Copyright GoldSim Technology Group LLC, 1998-2018. All rights reserved. GoldSim is a registered trademark of GoldSim Technology Group LLC.

Version 12.1 (June 2018)


THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

GoldSim Technology Group
22500 SE 64th Place, Suite 240
Issaquah, Washington 98027 USA

Visit us at our web site: www.goldsim.com
Email us at: software@goldsim.com
Contents

Chapter 1: Welcome to GoldSim! 1

Chapter Overview ........................................................................................................... 1
In this Chapter .................................................................................................................. 1
What is GoldSim? ........................................................................................................... 2
What Can I Do With GoldSim? ....................................................................................... 2
What Kind of Problems Can I Apply It To? ................................................................. 2
What Makes GoldSim Unique? ...................................................................................... 3
What Do I Need to Use GoldSim? .................................................................................. 4
How to Use this Manual ............................................................................................... 4
Example Models ............................................................................................................ 5
How this Manual is Organized ...................................................................................... 5
Conventions Used in this Manual ................................................................................. 7
Installing and Activating GoldSim .............................................................................. 7
Using a Standalone License ......................................................................................... 8
Using a Network License .............................................................................................. 13
Testing the GoldSim Installation .................................................................................. 17
activating and Deactivating Extension Modules ......................................................... 18
Sharing a License Between GoldSim Versions on a Computer ........................................ 19
Learning to Use GoldSim ............................................................................................ 19
Using GoldSim’s Help System .................................................................................... 21
Technical Support, User Resources and Software Upgrades ...................................... 21
GoldSim Maintenance Program .................................................................................... 21
Getting Technical Support ......................................................................................... 21
Other GoldSim Resources ............................................................................................ 22

Chapter 2: GoldSim in a Nutshell 23

Chapter Overview .......................................................................................................... 23
In this Chapter ................................................................................................................ 23
Understanding Simulation ............................................................................................... 24
Dynamic Simulation ....................................................................................................... 24
Probabilistic Simulation ................................................................................................. 24
Steps in Carrying Out a Simulation .............................................................................. 25
The Power of Simulation ................................................................................................. 26
What is GoldSim? ........................................................................................................... 27
A Powerful, Flexible Simulator ...................................................................................... 28
A System Integration Tool .............................................................................................. 28
A Visual Information Management System .................................................................... 28
Basic GoldSim Concepts ............................................................................................... 29
The GoldSim Simulation Environment .......................................................................... 29
Elements: The Basic Building Blocks in GoldSim .................................................... 29
Linking Elements .......................................................................................................... 31
A Simple Example ......................................................................................................... 31
Understanding Dynamic Simulation ........................................................................... 31
GoldSim is Dimensionally-Aware ................................................................................ 34
Representing Uncertainty .............................................................................................. 35
Representing Feedback Loops ...................................................................................... 36
Simulating Delays .......................................................................................................... 38
Building Hierarchical Top-Down Models .................................................................... 39
Additional Function Elements

Advanced Features

Manipulating Arrays (Vectors and Matrices)
Modeling Discrete Events
Activating and Deactivating Portions of a Model
Controlling the Timestep in a Model
Carrying Out Iterative (Looping) Calculations
Dynamically Linking to Spreadsheets
Importing Inputs from a Database
Building Custom Elements Using Scripts
Dynamically Linking to External Models
Building Large, Complex Models
Modeling Scenarios
Optimizing a Model
Carrying Out Sensitivity Analyses on a Model

Features for Documenting and Presenting Your Model

Creating Report Quality Result Graphics
Internally Documenting Your Model
Using GoldSim as a Presentation Tool

Specialized GoldSim Modules

Financial Module
Contaminant Transport Module
Reliability Module
Dashboard Authoring Module
Distributed Processing Module

The GoldSim Player

Chapter 3: Building a Model in GoldSim

Chapter Overview

In this Chapter

The GoldSim User Interface

The GoldSim Start Dialog
User Interface Components
Types of GoldSim Objects
Common Mouse Actions in GoldSim
Saving, Opening, and Closing GoldSim Files
Restoring Files After an Unexpected Failure Using Auto-Save
Password-Protecting a Model File
Sending Your Model to Someone Via Email
Simulation Modes

Creating Elements and Links

GoldSim Element Types
Creating Elements
Element Inputs and Outputs
Editing an Element’s Properties and Creating Links
Understanding Output Attributes
Using Dimensions and Units
Error Checking in Input Fields
Creating Links Using the Link Cursor
Understanding Containers
Understanding Influences
Referencing Time in GoldSim
Copying, Moving, and Deleting Elements
Navigating and Viewing a Model

Navigating Within the Graphics Pane
Chapter 4: Using the GoldSim Elements 123

Overview of GoldSim Element Types .................................................. 146
Using Basic Input Elements .............................................................. 151
Using Time Series Elements ............................................................ 186
Using Stock Elements ....................................................................... 228
Using Basic Function Elements ........................................................ 274

Running a Model and Viewing Results ............................................. 115
Specifying Simulation Settings ......................................................... 115
Saving Results .................................................................................. 116
Running a Model ............................................................................. 117
Viewing Results .............................................................................. 118
Building and Running Your First Model ........................................... 119

Chapter Overview ............................................................................ 123
In this Chapter ................................................................................ 123

Entering Mathematical Expressions into Element Input Fields .......... 124
Built-in Functions ............................................................................ 124
Built-in Constants .......................................................................... 129
Creating Conditional Expressions ................................................... 130
Referencing Dates in Expressions ................................................... 131

Using Containers ............................................................................. 132
The Container Properties Dialog ....................................................... 132
Container Options and Features ....................................................... 133
Controlling the Appearance of the Graphics Pane in a Container ...... 139
Summary Information for a Container .............................................. 140
Controlling Result Flags for Elements in the Container .................. 141
Sealing and Locking Containers ....................................................... 143

Using Data Elements ....................................................................... 146
Stock Elements .............................................................................. 147
Function Elements .......................................................................... 147
Event Elements ............................................................................... 148
Delay Elements ............................................................................... 149
Result Elements ............................................................................. 149
Differentiating Between Material and Information Flow ................. 150

Using Integrator Elements ............................................................... 188
Reservoir Elements .......................................................................... 188
Specifying What the Input to a Time Series Represents ................... 191
Viewing and Editing Time Series Inputs ........................................... 190
Specifying Time Series Outputs ...................................................... 200
Referencing a Time Series Using a Function .................................... 203
Time Series Examples ...................................................................... 204
Advanced Time Series Options ........................................................ 213
Browser View of a Time Series Element .......................................... 227
Chapter 6: Customizing the Interface

Chapter Overview ................................................................. 421
In this Chapter .................................................................. 421
Customizing Toolbars ......................................................... 422
Activating and Deactivating Toolbars .......................................................... 422
Moving and Docking Toolbars and the Menu Bar ....................................... 423

Customizing the Appearance of the Graphics Pane ..................................... 423
The Graphics Pane Grid and Background .................................................. 423
Adjusting the Size of the Graphics Pane ...................................................... 425
Saving the Graphics Pane's Position and Scale ........................................... 426
Editing the Appearance of Influences ......................................................... 426
Filtering Influences ....................................................................................... 430
Copying Container Settings to Other Containers in a Model ....................... 432

Editing the Appearance of Elements ............................................................ 434
Changing the Element's Symbol .................................................................. 434
Changing the Element's Label ..................................................................... 436
Changing the Element's Background and Outline ....................................... 437
Limiting the Changes That Can Be Made to the Element's Appearance .... 438

Viewing and Creating Units ........................................................................ 439
The GoldSim Units Manager ......................................................................... 439
Creating New Units ...................................................................................... 441
Using Placeholders for Time Units .............................................................. 445
Using the Percentage Unit Symbol .............................................................. 446

The Options Dialog ....................................................................................... 446
The General Tab of the Options Dialog ....................................................... 446
The Graphic Tab of the Options Dialog ......................................................... 447
The Results Tab of the Options Dialog ........................................................ 448

Chapter 7: Running a Model ......................................................................... 451

Chapter Overview .......................................................................................... 451
In this Chapter ............................................................................................... 451

Simulation Settings ....................................................................................... 452
Setting the Basic Time Options .................................................................... 453
Advanced Timestep Options .......................................................................... 466
Setting the Monte Carlo Options .................................................................. 479
Defining and Referencing Global Properties ............................................... 484
Viewing and Editing Model Summary Information ...................................... 485

Understanding and Referencing Run Properties ......................................... 486
Run Properties: Calendar Time ..................................................................... 488
Run Properties: Elapsed Time ...................................................................... 490
Run Properties: Simulation ......................................................................... 491
Run Properties: Reporting Periods ............................................................... 493

Saving Outputs as Results ........................................................................... 494
Specifying Results to be Saved .................................................................... 494
Highlighting Outputs that Will be Saved ...................................................... 497
Archiving a File with Results ....................................................................... 497

Running and Viewing the Status of a Simulation ......................................... 497
Understanding Simulation Modes .................................................................. 498
Carrying Out a Simulation (Run Mode) ......................................................... 499
Viewing Results (Result Mode) ..................................................................... 504

Creating, Running and Comparing Scenarios .............................................. 504
Introduction to Scenarios ............................................................................. 505
Creating and Editing Scenarios Using the Scenario Manager ...................... 513
Browsing and Editing the Active Scenario .................................................. 516
Running Scenarios and Displaying Scenario Results .................................. 520
Creating and Editing Scenarios in Dashboards ............................................ 527

Running an Optimization ............................................................................ 528
Overview of Optimization ........................................................................... 528
Defining the Optimization Settings .............................................................. 529
Chapter 9: Documenting and Presenting Your Model

Copying and Exporting Result Displays ....................................................... 734
Copying a Chart or Table .................................................................. 734
Exporting a Chart ............................................................................. 734

Adding Graphics and Text ........................................................................ 742
Adding Graphic Objects ...................................................................... 743
Changing the Appearance of Graphic Objects ..................................... 746
Adding and Editing Text ...................................................................... 747
Adding and Editing Text Boxes .............................................................. 751
Adding Images ..................................................................................... 753

Modifying the Appearance of Elements .................................................. 755

Creating, Editing and Viewing Notes ...................................................... 755
Opening the Note Pane .......................................................................... 755
Creating and Displaying Notes ............................................................... 758
Editing and Formatting Notes ................................................................. 759
Inserting Hyperlinks into Notes ............................................................... 759

Adding Hyperlinks to the Graphics Pane .................................................. 760
Specifying Addresses for the Various Hyperlink Types ......................... 762
Changing the Appearance of a Hyperlink Object .................................. 763

Manipulating Graphical Objects .............................................................. 767
Aligning and Ordering Objects ................................................................. 767
Spacing and Sizing Objects ................................................................... 768
Precisely Moving Objects ................................................................... 769
Grouping Objects ............................................................................... 769
Rotating Objects ................................................................................ 769
Copying, Pasting, Moving and Deleting Graphical Objects .................. 770
Using the Graphical Undo and Redo Functions .................................... 771

Creating Printed Documentation ......................................................... 771
Printing the Graphics Pane .................................................................. 771
Exporting the Graphics Pane .............................................................. 772
Creating an XML Model Inventory File ............................................... 772

Creating a GoldSim Player File .............................................................. 777
Creating a Player File Using the Dashboard Authoring Module ............... 779

Chapter 10: Advanced Modeling Concepts .................................................. 781

Chapter Overview ................................................................................ 781
In this Chapter ...................................................................................... 781

Using Vectors and Matrices .................................................................. 782
Understanding Array Labels ................................................................ 783
Defining Vectors and Matrices Using Data Elements ......................... 790
Defining Vectors Using Stochastic Elements ...................................... 795
Defining Arrays in an Input Field Using Array Constructor Functions ... 795
Using a Vector as a Lookup Table ....................................................... 798
Manipulating Vectors and Matrices with Other Elements ................... 799
Viewing Results for Arrays ................................................................ 807
Copying Array Elements Between Models ......................................... 807

Understanding Locally Available Properties ........................................ 808

Modeling Aging Chains ....................................................................... 810
Modeling Aging Chains Using a Series of Reservoirs or Pools .......... 810
Modeling Aging Chains Using a Series of Material Delays ................. 812
Modeling Aging Chains Using Integrators with Discrete Pushes ......... 814

Solving Convolution Integrals .............................................................. 817
What is a Convolution Integral? .......................................................... 817
Using the Convolution Element ......................................................... 818
Examples of the Use of the Convolution Element .............................. 821

Generating Stochastic Time Histories .................................................. 826
Types of Stochastic Time Histories ..................................................... 828
Generating Geometric Growth Histories ........................................... 830
Generating Random Walk Histories .................................................. 834
Simulating Correlated Arrays of Stochastic Histories ......................... 837

Using Resources ................................................................................ 839
Defining Resource Types and Creating Resource Stores ................. 839
Interacting with Resources ................................................................ 848
Summarizing Resource Locations and Users ..................................... 855
Viewing Resource Results ................................................................. 859

Script Elements .................................................................................. 860
Getting Started with the Script Element ............................................. 862
Controlling Program Flow in the Script Element ............................... 874
Understanding Variable Scope in a Script ......................................... 885
Editing Scripts ................................................................................... 887
Documenting Scripts ......................................................................... 890
Debugging Scripts ............................................................................. 890
Logging Messages in Scripts ............................................................. 892
Printing Scripts ................................................................................. 894
Script Examples ................................................................................. 895
Appendix A: Introduction to Probabilistic Simulation 1043

Appendix Overview ................................................................................................................. 1043
In this Appendix ....................................................................................................................... 1043
Types of Uncertainty ............................................................................................................... 1044
Quantifying Uncertainty ......................................................................................................... 1044
Understanding Probability Distributions .............................................................................. 1044
Characterizing Distributions ................................................................................................. 1046
Specifying Probability Distributions ....................................................................................... 1047
Correlated Distributions ......................................................................................................... 1048
Variability and Ignorance ......................................................................................................... 1049
Propagating Uncertainty ........................................................................................................ 1050
A Comparison of Probabilistic and Deterministic Simulation Approaches ....................... 1051
References ............................................................................................................................... 1053

Appendix B: Probabilistic Simulation Details 1055

Appendix Overview ................................................................................................................. 1055
In this Appendix ....................................................................................................................... 1055
Mathematical Representation of Probability Distributions ................................................ 1056
Distributional Forms .............................................................................................................. 1056
Representing Truncated Distributions ................................................................................... 1069
Correlation Algorithms ......................................................................................................... 1070
Sampling Techniques ............................................................................................................ 1072
Generating Random Numbers to Sample Elements .............................................................. 1072
Latin Hypercube Sampling ...................................................................................................... 1077
Importance Sampling .............................................................................................................. 1079
Representing Random (Poisson) Events .............................................................................. 1083
Computing and Displaying Result Distributions ................................................................. 1084
Displaying a CDF .................................................................................................................... 1084
Displaying a PDF ..................................................................................................................... 1085
Computing and Displaying Confidence Bounds on the Mean .............................................. 1086
Computing and Displaying Confidence Bounds on CDFs and CCDFs .............................. 1086
Computing the Conditional Tail Expectation ........................................................................ 1088
Computing Sensitivity Analysis Measures .......................................................................... 1089
References ............................................................................................................................... 1094

Appendix C: Implementing External (DLL) Elements 1095

Appendix Overview ................................................................................................................. 1095
In this Appendix ....................................................................................................................... 1095
Understanding External (DLL) Elements ............................................................................ 1096
Implementing an External Function .................................................................................... 1096
Important Restrictions .......................................................................................................... 1096
External Function Format ....................................................................................................... 1097
The Input and Output Argument Arrays ............................................................................. 1099
External Function Examples ................................................................................................. 1102
External Function Calling Sequence ..................................................................................... 1105
Before the Simulation ............................................................................................................ 1106
During Each Realization ....................................................................................................... 1106
Before Each Realization ....................................................................................................... 1106
After Each Realization .......................................................................................................... 1106
After the Simulation ............................................................................................................. 1106
DLL Calling Details .............................................................................................................. 1107
64-Bit DLL Support .............................................................................................................. 1107
Returning Error Messages from External Functions ............................................................ 1107
Appendix D: GoldSim Units Database

Appendix Overview .......................................................................................................................... 1109
Built-in Units and Conversion Factors .......................................................................................... 1110

Appendix E: Database Input File Formats

Appendix Overview .......................................................................................................................... 1121
In this Appendix ............................................................................................................................... 1121
Creating a Generic Database ........................................................................................................ 1122
Creating a Simple GoldSim Database ............................................................................................ 1122
  Parameter Table ............................................................................................................................ 1122
  Parameter Reference Table ......................................................................................................... 1126
  Array Values Table ...................................................................................................................... 1127
  Probability Value Pairs Table ..................................................................................................... 1127
  Example File and Database Template ......................................................................................... 1127
Creating an Extended GoldSim Database ...................................................................................... 1128
  Parameter Table ............................................................................................................................ 1128
  Parameter Value Table .................................................................................................................. 1130
  Value Component Table .............................................................................................................. 1131
  Example File ................................................................................................................................ 1133

Appendix F: Integration Methods and Timestepping Algorithm

Appendix Overview .......................................................................................................................... 1135
In this Appendix ............................................................................................................................... 1135
Factors Affecting the Accuracy of Simulation Models .................................................................... 1136
Primary Numerical Approximations in GoldSim .......................................................................... 1137
  GoldSim Numerical Integration Algorithm .................................................................................. 1137
  Approximate Solutions to Coupled Equations .......................................................................... 1139
  Selecting the Proper Timestep .................................................................................................... 1140
Summary of GoldSim’s Dynamic Timestepping Algorithm ............................................................ 1140
  Defining Specific Periods with Shorter Timesteps ..................................................................... 1140
  Dynamically Adjusting the Timestep ......................................................................................... 1141
  Assigning Different Timesteps to SubSystems ......................................................................... 1141
  Accurately Simulating Discrete Events that Occur Between Timesteps ................................... 1141
  Using Advanced Algorithms to Solve Coupled Equations ....................................................... 1142

Glossary of Terms ............................................................................................................................ 1143

Index ................................................................................................................................................ 1155
Chapter 1: Welcome to GoldSim!

If a man will begin with certainties, he shall end in doubts, but if he will be content to begin with doubts, he shall end in certainties.

Francis Bacon, The Advancement of Learning

Chapter Overview

GoldSim is a user-friendly, highly graphical program for carrying out dynamic, probabilistic simulations to support management and decision-making in business, engineering and science.

This User's Guide provides a complete description of the features and capabilities of GoldSim.

In this Chapter

This introductory chapter discusses the following topics:

- What is GoldSim?
- How to Use this Manual
- Conventions Used in this Manual
- Installing and Activating GoldSim
- Learning to Use GoldSim
- Using Help and the GoldSim Tutorial
- Getting Technical Support
What is GoldSim?

GoldSim is a highly graphical, object-oriented computer program for carrying out dynamic, probabilistic simulations. In a sense, GoldSim is like a "visual spreadsheet" allowing you to visually create and manipulate data and equations.

What Can I Do With GoldSim?

GoldSim is a simulation program. As used here, simulation is defined as the process of creating a model (i.e., an abstract representation or facsimile) of an existing or proposed system (e.g., a business, a mine, a watershed, a forest, the organs in your body, the atmosphere) in order to identify and understand those factors which control the system and/or to predict (forecast) the future behavior of the system. Almost any system which can be quantitatively described using equations and/or rules can be simulated.

Simulation is an important tool because it provides a way in which alternative designs, plans and/or policies can be evaluated without having to experiment on a real system, which may be prohibitively costly, time-consuming, or simply impractical to do. That is, simulation allows you to ask "What if?" questions about a system without having to experiment on the actual system itself (and hence incur the costs and delays associated with field tests, prototypes, etc.).

Because GoldSim was designed with flexibility in mind, you can use it to simulate almost any kind of system. Illustrative examples from the fields of business, science and engineering include the following:

**Strategic Planning:** You could simulate the implementation of a complex undertaking (e.g., design, manufacture and marketing of a new product) by describing the tasks involved, any precedent requirements (i.e., what must be done before a particular task can begin or end), task durations and costs, and events which could impact the process. The output of such a simulation might be the probability of successfully completing the undertaking (in a certain timeframe, or at a certain level of profitability). You could use the results to identify potential problems that might arise and design contingency plans. On a larger scale, such a tool could be then used to evaluate and manage portfolios of projects and investments.

**Ecology:** The growth of a group of animals could be simulated by describing in mathematical terms the initial number of animals, the birth
rate, the death rate, the rate at which animals migrate to or away from the group, possible catastrophic events, etc. The output of this simulation would then be the number of animals in the group as a function of time (e.g., one year from now, ten years from now, etc.). You could use the results to better manage the system in order to stabilize or increase the population (e.g., by limiting hunting, or introducing predators).

**Environment:** You could simulate the performance of a hazardous waste site by describing the initial conditions (e.g., the geometry of the system, the amount of contaminants in the system) and the processes acting on the system (e.g., degradation of the drums containing the waste, migration of contaminants through the environment). The output of this kind of simulation would be contaminant concentrations around the site as a function of time. You could use the results to design remediation measures which would minimize the environmental impacts at the site.

**Reliability Engineering:** The reliability of a proposed satellite system could be simulated by describing the components of the system and the processes and events which could compromise the system’s integrity and lead to failure or downtime. The outputs of this kind of simulation would include the predicted reliability of the system and the probability and consequences of different types of failures. You could use the results to modify the design so as to maximize the reliability and minimize the probability and/or consequences of a failure.

**Manufacturing:** You could simulate the coupled dynamics of a manufacturing supply chain by defining the “links” in the chain (Retailer, Distributor, Manufacturer, Tier 1 supplier(s), Tier 2 suppliers, etc.) and how these organizations interact with each other. The model would simulate the movement of materials (parts to finished product) through the supply chain, and could be used to identify ways in which the system could be modified (e.g., via technology or improved decision rules) to operate more efficiently.

GoldSim is user-friendly and highly graphical, such that you can literally draw and subsequently present a picture (or influence diagram) of the system you wish to model in an intuitive way without having to learn a great deal of symbols, notation and functions.

Because simulation can be such a powerful tool for understanding and managing complex systems, a variety of graphical simulation tools currently exist. The following combination of features, however, makes the GoldSim approach unique:

**GoldSim was specifically designed to quantitatively address the inherent uncertainty which is present in real-world systems.** GoldSim provides powerful tools for representing uncertainty in processes, parameters and future events, and for evaluating such systems in a computationally efficient manner.

**GoldSim provides powerful capabilities for superimposing the occurrence and consequences of discrete events onto continuously varying systems.** This allows for the realistic simulation of discrete events such as financial transactions, accidents, system failures, storms, labor strikes, and lawsuits.

**GoldSim was designed to facilitate the construction of large, complex models.** You can build a model of your system in a hierarchical, modular manner, such that the model can readily evolve and add detail as more knowledge regarding the system is obtained. Other powerful features, such
as the ability to manipulate arrays, the ability to “localize” parts of your model, and the ability to assign version numbers to a model which is constantly being modified and improved, further facilitate the construction and management of large models.

**GoldSim is dimensionally-aware.** GoldSim has an extensive internal database of units and conversion factors. You can enter data and display results in any units. You can even define your own customized units. GoldSim ensures dimensional consistency in your models and carries out all of the unit conversions internally. As a result, when you use GoldSim, it is never necessary for you to carry out (error-prone) unit conversions.

**GoldSim is highly extensible.** You can dynamically link external programs or spreadsheets directly into your GoldSim model. In addition, GoldSim was specifically designed to support the addition of customized modules (program extensions) to address specialized applications.

**GoldSim allows you to create compelling presentations of your model.** A model that cannot be easily explained is a model that will not be used or believed. GoldSim was specifically designed to allow you to effectively document, explain and present your model. You can add graphics, explanatory text, notes and hyperlinks to your model, and organize it in a hierarchical manner such that it can be presented at an appropriate level of detail to multiple target audiences.

The program runs on personal computers using 32-bit or 64-bit Microsoft Windows® operating systems. Specifically, Windows 7 (either 32-bit or 64-bit versions), Windows 8, Windows 8.1 and Windows 10 are the supported operating systems. If you are linking GoldSim to Microsoft Excel, you must have Excel 2003 or later (2010 or later recommended). You must have Administrative Privileges on the system during the installation and a minimum of 500 MB disk space. 32-bit color depth is required.

Because GoldSim is very powerful and flexible, it is relatively complex. Nevertheless, the software can be readily mastered by anyone familiar with the basic functions of a personal computer and the Windows operating system. Since a GoldSim model is built by describing functional relationships (equations) between the components of your system, however, you must be comfortable with the basics of quantitative analysis. This does not mean you must be a mathematician or a numerical modeler. As a rule of thumb, if you are comfortable using a spreadsheet you can learn to build a model in GoldSim.

Finally, although GoldSim can be run in a deterministic manner (i.e., with no uncertainty specified in the input parameters), one of the key features of the program is its ability to explicitly represent such uncertainty through the use of probability distributions. In order to do so, you must have at least a basic understanding of the representation and propagation of uncertainty. Appendix A provides a brief primer on this topic, along with suggestions for further reading.

**How to Use this Manual**

One way to learn a new program is to read the user’s guide (if it exists and is well-written) from front to back, while simultaneously experimenting with the program. Although a few people may prefer this method, we realize that most software users (including ourselves!) rarely have the time or patience to do this. Instead, most of us prefer to read some introductory material (or take a training course), and then "dive in", referring to the documentation when we are stuck or need to learn a new feature. Therefore, this document is designed primarily to be used in this way, as a reference document.
Note: As will be discussed at the end of this chapter, because it can be time-consum ing and inefficient to learn how to use a new software tool by only referring to detailed reference documentation, a number of other resources are available that complement this document and are specifically intended to help beginners learn to use GoldSim.

Read more: Learning to Use GoldSim (page 19).

The document is heavily cross-referenced, such that each chapter and each major section can stand alone. To facilitate the use of these cross-references, whenever they appear in the manual, they are set off from the text and always proceeded by “Read More:”), as shown above.

This allows the document to be readily used as a reference guide, with the user accessing topics and features via the cross-references, the table of contents and/or the index on an as-needed basis. Furthermore, as will be discussed later in this introduction, nearly all of the information available in the printed documentation can also be accessed via an extensive help system.

Note: Although this manual is written primarily so that it can be used as a reference document, it is highly recommended that all users read Chapters 2 and 3 before starting to use GoldSim.

Example Models

Starting with Chapter 3, the manual references a number of examples that are relevant to features being discussed. When you install GoldSim, a number of examples are installed with the program. These example model files are mentioned in the text, and provide an excellent way to begin to experiment with GoldSim. Each example is internally documented with comments regarding the way the model was implemented.

Note: You can quickly access these files by selecting File|Open Example… from the main GoldSim menu.

Note: These examples are read-only files. If you wish to modify them, you will need to Save As.

Note: Additional (and generally more detailed) example models are available in the Model Library on the GoldSim web site.

How this Manual is Organized

This document is organized into ten chapters:

Chapter 1: Welcome to GoldSim! The remainder of this chapter discusses the information required for you to get started using GoldSim, including conventions used in the manual, installing the program, using online help, and obtaining technical support.
All users should read Chapters 2 and 3! These two chapters will provide you with enough information to start using GoldSim.

Chapter 2: GoldSim in a Nutshell. This chapter provides a broad overview of the features and capabilities of GoldSim. It is strongly recommended that you read this chapter, as it will tell you what the program is capable of doing, and direct you to those portions of the manual where you can obtain further information.

Chapter 3: Building a Model in GoldSim. This chapter describes the GoldSim user interface, and presents all of the techniques required for creating, editing, and navigating a GoldSim model. Before trying to experiment with GoldSim, it is highly recommended that you read this chapter in order to familiarize yourself with these basic techniques.

Chapter 4: Using the GoldSim Elements. GoldSim models are built using a wide variety of model objects (referred to as elements). This chapter describes the details of using the basic elements in GoldSim.

Chapter 5: Simulating Discrete Events. In many systems, processes occur that are discrete as opposed to continuous. These discrete occurrences are referred to within GoldSim as events. This chapter describes the GoldSim elements that you can use to simulate the occurrence and consequences of discrete events.

Chapter 6: Customizing the Interface. This chapter provides instructions for customizing the appearance of the user interface to meet your specific needs.

Chapter 7: Running a Model. After you have created a model, you need to run the model in order to produce results. This chapter describes the details of how to control your simulation (e.g., specify its duration), specify the types of results you wish to save, and run the model.

Chapter 8: Displaying Results in GoldSim. This chapter describes how you can view (via charts and tables), analyze, and present model results.

Chapter 9: Documenting and Presenting Your Model. GoldSim was specifically designed to allow you to effectively document, explain and present your model. This chapter describes how you can add graphics, explanatory text, notes and hyperlinks to your model, and organize it in a hierarchical manner such that it can be presented at an appropriate level of detail to multiple target audiences.

Chapter 10: Advanced Modeling Concepts. This chapter describes advanced and powerful capabilities of GoldSim that you will want to take advantage of once you become comfortable with the program, including using and manipulating arrays (vectors and matrices) in your models, simulating discrete events, and linking GoldSim to a database.

This manual also includes six appendices.

Appendix A: Introduction to Probabilistic Simulation. This appendix provides an introduction to basic concepts of probabilistic simulation, and provides suggestions for further reading.

Appendix B: Probabilistic Simulation Details. This appendix describes the mathematical details of the manner in which uncertainty is represented, propagated, and displayed in GoldSim.

Appendix C: Implementing External (DLL) Elements. This appendix provides instructions for users who wish to directly link external programs into a GoldSim model.
Appendix D: GoldSim Units Database. As mentioned above, GoldSim is dimensionally aware and carries out all unit conversions internally. This appendix lists all of the internal units and conversion factors provided by GoldSim.

Appendix E: Database Input File Formats. GoldSim allows data to be imported from an ODBC-compliant database directly into GoldSim. This appendix describes the required database structures and formats to facilitate such a data transfer.

Appendix F: Integration Methods and Timestepping Algorithm. This appendix describes the methods used by GoldSim to numerically integrate differential equations. It also discusses the unique timestepping algorithm GoldSim uses to accurately represent discrete events.

The manual also includes a Glossary of Terms and an Index.

Conventions Used in this Manual

The following conventions are used in this manual:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Important Terms</td>
<td>New and important terms are presented in <strong>bold italics</strong>. These terms all appear in the Glossary of Terms at the end of the document.</td>
</tr>
<tr>
<td>File</td>
<td>Open...</td>
</tr>
<tr>
<td>CTRL+C</td>
<td>Key combinations are shown using a “+” sign. <strong>CTRL+C</strong> means press the Control and C keys simultaneously.</td>
</tr>
<tr>
<td>Warning:</td>
<td>This means watch out! Warnings typically alert you to potential pitfalls and problems that may occur if you perform (or fail to perform) a certain action.</td>
</tr>
<tr>
<td>Note:</td>
<td>Notes highlight important information about a particular concept, topic or procedure, such as limitations on how a particular feature can be used, or alternative ways of carrying out an action.</td>
</tr>
</tbody>
</table>

In describing various mouse actions, the following conventions are used:

<table>
<thead>
<tr>
<th>Mouse Action</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click</td>
<td>Press and release the left mouse button once.</td>
</tr>
<tr>
<td>Double-click</td>
<td>Press and release the left mouse button twice in rapid succession.</td>
</tr>
<tr>
<td>Right-click</td>
<td>Press and release the right mouse button once.</td>
</tr>
<tr>
<td>Drag</td>
<td>Press the left mouse button, and while keeping it depressed, move the cursor to another location, then release the button.</td>
</tr>
</tbody>
</table>

Installing and Activating GoldSim

To install GoldSim, double-click the GoldSim installation file and follow the directions on the screen. The installation file for the latest version of GoldSim (as well as previous versions) is available on the GoldSim web site.
**Note:** In order to install GoldSim, you must have Administrator privileges on the computer.

During the installation process, the GoldSim License Agreement will be displayed in a dialog.

You can press the **Print…** button to print the document at this time to obtain a hardcopy version of the License Agreement. You must select “I accept the terms in the license agreement” to continue with the installation.

**Note:** After completing the installation of GoldSim, you can access the License Agreement at any time by selecting **Help|About GoldSim** from the main menu and pressing the **User License Agreement** link on that dialog.

Once the program is installed, GoldSim can be used by either activating a **Standalone license** on your computer, or by accessing a **Network license** that has been activated on a License Server.

Standalone licenses are activated on a single computer and, once they are activated, do not require connection to the internet (the license resides on the computer). Network licenses reside on a License Server, and in order to use the license, your computer must be persistently connected to the License Server (although as will be discussed below, licenses can be “borrowed” from the License Server so that they temporarily reside on your computer and a connection to the License Server is no longer necessary).

If you are unsure of what type of license you have, refer to the email you received from the GoldSim Technology Group when the license was provided (this may have been sent to a License Administrator).

There are two types of **Standalone licenses**: Desktop Standalone licenses, and Enterprise Standalone licenses. The licenses are identical in all respects with one exception: Desktop Standalone licenses limit the number of license transfers (to 6 per year). Enterprise Standalone licenses allow an unlimited number of transfers.

This section describes how to activate a GoldSim **Standalone License** (Desktop or Enterprise). The instructions below assume that you have an internet connection on the computer where you have installed GoldSim. If you do not have an internet connection, contact your License Administrator or GoldSim Technology Group for instructions on how to activate the license manually.

When you open an unlicensed copy of GoldSim, you will see the following dialog:
To activate a Standalone license, press the **Activate Standalone License** button. The following dialog will then be displayed:

An **Activation ID** should have been provided to you (or your License Administrator) via email. The Activation ID is a 32-digit code (8 groups of 4 digits each, separated by hyphens) that is used to activate your license. To avoid errors, rather than typing the Activation ID in by hand, it is suggested that you copy your Activation ID (highlight it and press **Ctrl+C**) from the email that you received and paste it (**Ctrl+V**) into the **Activation ID** field.

After doing so, click the **Activate** button. You will momentarily see a progress dialog while GoldSim carries out the activation. If your license activation was successful, you will see the following message dialog:

When you click **OK**, GoldSim should open.

After successful activation, you can view your license details by selecting **Help|Licensing...** from the main GoldSim menu. This will open the GoldSim License Manager:
In this dialog, you can see details about your license. The middle section, *Local License Management*, displays your *Activation ID*, the *Product name*, the *Latest Supported Version*, the *Licensee* organization and the *Expiration* date of the license.

You can use any version of GoldSim including and prior to the *Latest Supported Version* with your GoldSim license. Note, however, that the current licensing system only supports GoldSim 10.5 (released in 2010) and later. If you need to use a very old version of GoldSim (previous to 10.5), contact the GoldSim Technology Group.

Although GoldSim licenses are perpetual, they still expire (typically after one year) and must be re-activated annually (at no charge). GoldSim will send reminders to your License Administrator several months prior to the *Expiration* date listed in the dialog.

If you ever need to provide your Activation ID to GoldSim Technology Group or to someone within your organization, you can copy the Activation ID to the clipboard using the copy button in the GoldSim License Manager dialog located to the immediate right of the Activation ID:

If you have already activated GoldSim on a computer and you wish to transfer the license to another computer, you can return (i.e. deactivate) the license and then activate it on another computer. This can be done very quickly without having to contact GoldSim Technology Group. This allows you to easily share your license with others.

Be aware, however, that if you are using a Desktop Standalone license, you are limited in the number of times you can transfer your license (6 times per year). If you plan to frequently move a Standalone license between computers and you...
have a Desktop Standalone license, you can upgrade to an Enterprise Standalone license, which allows an unlimited number of transfers. This is useful in the case where you are frequently sharing the license with other users.

If you are unsure what type of license you have (Desktop or Enterprise), open GoldSim and from the main menu select Help|Licensing… to open the License Manager. In the Local License Management section, you can see information about your license. The license type is shown in parentheses in the Product line:

The instructions below for returning a license assume that you have an internet connection on the computer where you have installed GoldSim. If you do not have an internet connection, contact your License Administrator or GoldSim Technology Group for instructions on how to deactivate the license manually.

To return (deactivate) a Standalone license, open GoldSim and from the main menu select Help|Licensing… to open the License Manager:

From the drop-list in the Local License Management section, select the Activation ID for the license that you want to return (in most cases, there will only be one in the list).

Note: The Activation ID is a drop-list, since under some license use scenarios, you could have multiple activations on your computer (e.g., if you temporarily needed to borrow a license with additional modules that was available on a Network license server, but not on your Standalone license). In most situations, however, there will only be a single Activation ID in the list. The Fulfillment Details refer to the selected Activation ID, so you can easily understand what each ID represents.

After you return a license, in most cases, you will want to communicate to others in your organization (e.g., your License Administrator, a colleague who wants to use the license) which Activation ID you have returned. Therefore, before you return the license, you should copy the Activation ID to the
Installing and Activating GoldSim

clipboard so you can easily paste it into an email. You can do so by pressing the copy button to the right of the Activation ID:

Once the the Activation ID has been selected (and copied), click the Return… button. You will see a dialog asking you to confirm whether or not you want to proceed with the return. Note that his message indicates what type of Standalone license you have (Desktop or Enterprise):

It also prompts you to record the Activation ID (if you have not already done so). You cannot highlight and copy text in this dialog. Hence, if you have not already copied the Activation ID (as described above), the easiest thing to do is press No in this dialog, which will return you to the License Manager dialog. You can then copy the Activation ID using the copy button to the right of the Activation ID, and proceed again with the return.

If the return was successful, you will be notified. Once a license has been returned, it will no longer be used on the computer from which you returned the license. This means that you can activate that license on a different computer using the same Activation ID.

In addition to initially activating your Standalone license, there are three instances in which you will need to re registers the license (provide a new Activation ID):

- Although GoldSim licenses are perpetual, they still expire (typically after one year) and must be re-activated annually (at no charge). GoldSim will send reminders to your License Administrator several months prior to the Expiration date (listed in the GoldSim License Manager dialog).
When a new version of GoldSim is released that you are eligible to use (i.e., your license is under active maintenance).

In some cases, you may decide to upgrade your license (e.g., add a specialized extension module, or convert from a Desktop to an Enterprise Standalone).

**Read more:** Specialized GoldSim Modules (page 52).

In all of these instances, you will be provided with a new Activation ID by the GoldSim Technology Group. You will then need to reactivate your license with the new Activation ID by following these simple steps:

1. Select **Help|Licensing…** from the main GoldSim menu to open the GoldSim License Manager dialog.

2. Press the **Activate…** button in the GoldSim License Manager dialog to display this dialog:

   ![GoldSim License Manager Dialog](image)

   3. Enter the new Activation ID and press **Activate**.

**Using a Network License**

Network licenses allow you to easily share licenses across an organization. To use a Network license, you still must have GoldSim installed locally on your computer, but the license itself resides on a License Server.

In order to use the license, your computer must be persistently connected to the License Server (although as will be discussed below, licenses can be “borrowed” from the License Server so that they temporarily reside on your computer and a connection to the License Server is no longer necessary).

To use GoldSim by connecting to a Network license, your License Administrator must have already set up and activated the Network license and provided you with an address to the License Server.

When you open an unlicensed copy of GoldSim, you will see the following dialog:

![GoldSim Licensing Dialog](image)

To use a Network license, press the **Connect to Network License** button. You will see the Network License Configuration dialog:
Entering the License Server address provided to you by your License Administrator. If there are modules available on the License Server that you would like to use, you can request a module by checking the box next to the module you would like to use:

![Network License Configuration dialog]

**Note:** When connecting to a Network license, you will only acquire those modules that you specifically request here. If you do not request a module, you will not acquire it (even if it is available on the License Server).

If there is an available license on the License Server, GoldSim will open after you press **Close**.

**Note:** If a GoldSim license was available, but a requested module was not, GoldSim will still open but a warning message will inform you that the requested module(s) were not acquired.

If you subsequently open the License Manager dialog (by selecting **Help|Licensing...** from the main GoldSim menu), the dialog will look something like this:
The top section, *Acquired License Features*, will indicate that your license is a Network license. If you were successful in acquiring any modules you requested from the License Server, this will also be indicated in this section.

The bottom section (*Network License Configuration*) shows the Network License Address and also the features you have requested from the License Server. Note that this section indicates what you have requested from the License Server. This does not indicate what you have actually acquired from the License Server. If all licenses are in use or the License Server does not have a module you are requesting, you will get one or more error messages indicating you were not able to acquire the requested feature(s). You can always see what features you have successfully acquired by looking at the *Acquired License Features* section.

Note that you only need to enter the Network License Address (and requested features) once. After you do so, the next time you open GoldSim, it will use that same information to attempt to acquire a license from the License Server.

**Note:** Once you have acquired a license, you can change the License Server and the modules you wish to request by pressing the *Configure...* button, which reopens the Network License Configuration dialog.

### Borrowing a Network License for Offline Use

When using a Network license, you may occasionally need to use GoldSim while disconnected from the License Server (for example, when traveling). To facilitate this, GoldSim provides the option to **borrow** a license from a License Server for disconnected use. To initially borrow the license, you need to be connected to the License Server. But once the license is borrowed, you can disconnect from the License Server and use the license as though it were a local (i.e., Standalone) license.

At the time you borrow a license, you must specify a duration (in days) for the borrowing period. At the end of this period, the local license (on your computer) automatically expires and the license becomes available again on the License Server.

If you wish to borrow a license and are currently using a Network license, open the License Manager (by selecting **Help|Licensing...** from the main GoldSim menu). Within the License Manager dialog, press the **Borrow...** button.
If your copy of GoldSim is currently unlicensed (but you do have access to the License server) and you wish to directly borrow a license, press the **Borrow Network License** button on the dialog that is displayed when you try to run GoldSim.

In both cases, the following dialog will appear:

To borrow a license, you will need to enter a **License Server** address and the **Activation ID** for the license that you wish to borrow. You will need to get these from your License Administrator. You must also specify a **Duration** (in days) for which you wish to borrow the license. This can be no longer than 30 days (and may be further limited by your License Administrator).

After entering this information and pressing the **Borrow** button, you will see a progress dialog while GoldSim attempts to borrow the license from the License Server. This should typically only take a few seconds. If the borrow attempt is successful, you will see a message dialog that shows the License Server address and the date on which the borrowed license will expire:

Once you have borrowed a license, in the **Local License Management** section of the License Manager, you can see details for the borrowed license. The details are similar to what would be shown for a local (Standalone) license, with the following exceptions: (1) the **Product** line indicates that the license is a Network license and (2) the **Expiration** line indicates that the license is borrowed (and displays the expiration date):
If you wish to return the license before the expiration date, you can return (deactivate) the license by pressing the Return... button. When you do this, you will see a message dialog asking you to confirm the return action. (Note that to return the borrowed license early, you must have access to the License Server.)

After you confirm that you want to return the borrowed license, you will be prompted for the License Server address:

Once you have entered the License Server address, press Return. A progress dialog should appear while the return is carried out. If the return is successful, you will see a confirmation message.

Once GoldSim is activated, you should ensure that the program was properly installed. To test the GoldSim installation, do the following:

1. Double-click on the GoldSim program icon (on the desktop or, using Windows Explorer, in the folder into which you installed GoldSim) to start the program.
2. Press Esc to close the Start dialog that is presented.
3. From the main menu, select File | Open Example...
4. Enter the folder named "General Examples".
5. Select a file named "FirstModel.gsm" and press Open.
7. A dialog stating that the simulation has completed will be displayed. Press OK to close the dialog.
8. You will see an object labeled "Volume" on the screen.
9. Right-click the object. A menu will be displayed. At the top of the menu, you will see a Time History Result... option.
10. Click on this option. The following chart will be displayed:
11. If you are able to complete these steps and your chart looks like this, GoldSim has been properly installed. (Close the chart, and you can close GoldSim.)

Activating and Deactivating Extension Modules

GoldSim was designed to facilitate the incorporation of add-on modules (program extensions) to enable the program to address specialized systems. These modules add additional features and capabilities to GoldSim. Your license determines which of these program extensions are available to you.

Read more: Specialized GoldSim Modules (page 52).

You can view which modules are available by selecting **Model|Options...** from the main menu, and selecting the **Modules** tab:

All extension modules that you are licensed to use appear in the dialog. If your license does not allow you to use a particular module, it will not be listed. You can activate and deactivate modules that you are licensed to use by clicking the **Active** checkbox. By default, whenever you activate GoldSim, none of the available extension modules allowed by your license will be activated. To use them, you must activate them using this dialog.
If you deactivate a module, the specialized elements associated with that module will be deleted from your model (if any are present) and any menu options will be removed in the current file. If you make a module active, the various options associated with that module are made available again. If you press the **Set as Default** button, the selected modules will be activated for all new models that are created.

---

**Note:** If you try to open a file that contains elements associated with an extension module, and you are licensed to use that module but it is not currently active, GoldSim will automatically activate the module and open the file. If, however, you are not licensed to use the module, GoldSim will not open the file (and will display an error message).

---

### Sharing a License Between GoldSim Versions on a Computer

GoldSim allows you to automatically share a single license between different GoldSim versions on a single computer. This is particularly useful when converting files from an old version of GoldSim to a new one. In such a case, you may want to have two versions of GoldSim on a single machine (e.g., Version 10.5 and Version 12), and be able to switch back and forth between versions or run both versions simultaneously.

GoldSim allows you to easily share a license in this way. After installing a second version on your machine and trying to run it, GoldSim will automatically detect and use your previously existing license (even if you are running the two versions simultaneously).

### Learning to Use GoldSim

Although GoldSim’s intuitive interface will tempt you to simply dive in and start playing with the software, you are strongly discouraged from doing so, even if you are an experienced modeler. Spending time up front (by following the steps outlined below) is the most efficient way to understand the software’s features and capabilities and start building models in GoldSim.

1. **Take the GoldSim Tutorial.** The GoldSim Tutorial is by selecting **Help | Tutorial…** from the main GoldSim menu. The GoldSim Tutorial presents the basic concepts on which GoldSim is based and provides an overview of GoldSim’s key features and capabilities. After taking the Tutorial (which takes perhaps one or two hours), you will have sufficient understanding to start using the software, and will get more out of the information presented in the Help topics. This is the minimum that you must do before using GoldSim.

2. **Take the Online GoldSim Training Course.** Although the Tutorial is of value to quickly get an overview of GoldSim, it provides a very simple overview of only the most basic GoldSim concepts. The best and most effective way to learn GoldSim is to take a full training course. A free “hands-on” Online Training Course is available that will provide you with a thorough understanding of the key concepts on which GoldSim is based and all of the fundamentals required to build complex models of nearly any kind of system. That is, it does not simply focus on the mechanics of using the GoldSim software; just as importantly, it explains the fundamental concepts underlying dynamic, probabilistic simulation in general.

Because the Course is quite thorough, it will likely take as long as 40 hours to complete. Of course, if you are already somewhat familiar
with simulation (and/or have a strong quantitative background), you
may in fact be able to cover the material in considerably less than 40
hours.

You can take the Course at your own pace (and if you register, the site
will remember where you last left off). If you are new to simulation, the
ideal pace is probably to spend perhaps just a couple hours per day with
the Course. This then provides you with an opportunity to give some
considered thought to the information that was presented, and prevent
“information overload”.

Although taking the Course is time-consuming, over the long-term it is
by far the most efficient way to learn to use the software. Trying to
learn a complex and powerful software tool such as GoldSim on your
own by simply experimenting and consulting the reference material is
likely to take far in excess of 40 hours, and by doing so, you will likely
not discover many of GoldSim’s most powerful features.

You can find a link to the Course on the GoldSim website
(https://www.goldsim.com/Web/Customers/Education/Training/).

3. **Request your free one hour web-based training session.** When you
purchase GoldSim, you are entitled to a free one hour, live web-based
training session in which one of our analysts provides an interactive
training session via the Internet and telephone. You are strongly
encouraged to take advantage of this free training.

4. **Open and explore the Example Files.** When you install GoldSim, a
folder labeled "General Examples" is installed with the program. (You
can quickly access these files by selecting **File|Open Example…**
from the main GoldSim menu). The example model files show how to use
each of the elements (the examples are referenced in the Help system
and User’s Guide where the element is introduced). These example
model files are an excellent way to begin to experiment with GoldSim.
While exploring the files, use GoldSim’s context-sensitive Help (i.e.,
the **Help** button in each dialog) to access a detailed discussion of the
element.

5. **Download Example Files from the Model Library.** The GoldSim
website contains a Model Library with a number of models illustrating
how GoldSim can be used for particular applications. These models
 tend to be more complex than the simple example files found in the
General Examples folder, but still relatively simple. Again, while
exploring the files, use GoldSim’s context-sensitive Help (i.e., the **Help**
button in each dialog) to learn more about particular elements or
features utilized in the model.

6. **Browse the User’s Guide or Help System.** GoldSim has a large
number of features, and you will not discover all of them by
experimenting with simple example models. To fully utilize
GoldSim’s powerful features, browse through the User’s Guide, using
the index and table of contents as your guide. Each section of the
User’s Guide is heavily cross-referenced, so it is easy to just jump
around. Note that the Help system contains all of the contents of the
User’s Guide, with the exception of the technical appendices.

7. **Contact us with questions.** When you purchase GoldSim, you are
entitled to one year of free support. This does not include assistance in
building and debugging your models, but it does include answering
questions on how to use GoldSim’s features, so feel free to contact us!
Using GoldSim’s Help System

GoldSim has an extensive in-product help facility, which can be used to supplement this manual. The Help system can be accessed by selecting **Help** | **Help Topics** from the main GoldSim menu, pressing the **F1** key, or clicking on the Help button (the question mark) on the standard toolbar. Pressing the **Help** button (or **F1**) within any of the dialogs also provides access to Help (in a context-sensitive manner). All of the information in this document (with the exception of the appendices) is accessible via GoldSim Help.

Technical Support, User Resources and Software Upgrades

The GoldSim Technology Group is dedicated to providing complete solutions for our customers. We pride ourselves in providing prompt and extensive support and resources to our users, and are committed to ensuring that each installation of our software is successful and adds value to the customer.

When you purchase GoldSim software, you receive one year of Software Maintenance, entitling you to the following:

- Free software upgrades so that you always have the latest version of the GoldSim software.
- Basic Technical Support via email and phone. Basic support covers installation and licensing questions, as well as questions about GoldSim's features and capabilities.

After the first year, if you wish to continue to have access to new versions and technical support, Software Maintenance can be extended each year with payment of an annual fee.

Details regarding the GoldSim Maintenance Program can be found at www.goldsim.com/Web/Products/BuyGoldSim/Pricing/MaintenanceProgram/.

Users with active Software Maintenance can submit questions directly to the GoldSim support team. Evaluation users are also welcome to contact us with questions on GoldSim functionality. The **GoldSim Help Center** (https://goldsim.zendesk.com) is the primary portal for technical support. You can submit your questions directly from the Help Center. If your register and log in through the Help Center, you will be able check the status and view a history of all of your support requests.

The Help Center also includes:

- The **GoldSim Forum**, where you can you can post questions to the GoldSim community, or just browse existing messages;
- Articles on licensing questions and modeling tips; and
- An archive of past webinars (which demonstrate GoldSim features and capabilities).

Although using the Help Center is the preferred way to access support, you can also contact us by phone at 1-425-295-6985 between the hours of 8AM and 5PM Pacific Time.

Free Basic Technical Support does **not** include consulting, model troubleshooting or detailed assistance with applying GoldSim to a particular problem. Assistance of this nature is defined as Advanced Technical Support. Users may purchase Advanced Technical Support in pre-paid 10 hour blocks.
Details regarding Advanced Technical Support can be found at www.goldsim.com/Web/Resources/TechnicalSupport/.

In addition to the GoldSim Help Center, additional resources are also available. These three resources can be accessed directly from the GoldSim website (www.goldsim.com):

- A free Online Training Course that will provide you with a thorough understanding of the key concepts on which GoldSim is based and all of the fundamentals required to build complex models of nearly any kind of system.

- The GoldSim Model Library, which contains a collection of example models to allow you to see how specific features of GoldSim can be used and/or how GoldSim can be used for specific applications.

- The GoldSim Blog, which provides an informal mechanism for GoldSim staff to share their knowledge, point out some of the more advanced (and perhaps overlooked) GoldSim features, share and discuss common mistakes we see in GoldSim applications, discuss interesting applications, and keep you abreast of our plans for further GoldSim developments.

You can stay up to date on the latest GoldSim news through these resources:

- The GoldSim LinkedIn Group, which is primarily used for announcements (e.g., new versions, interesting applications). You can join the Group here: www.linkedin.com/groups/1798413

- Periodic email newsletters are sent two to three times per year. To be added to the newsletter list, contact us via the GoldSim Help Center (https://goldsim.zendesk.com).

Note: When you purchase GoldSim, you are entitled to a free one hour, live web-based training session in which one of our analysts provides an interactive training session via the Internet and telephone. You are strongly encouraged to take advantage of this free training.
Chapter 2: GoldSim in a Nutshell

The purpose of computing is insight, not numbers.

R.W. Hamming, *Numerical Methods for Scientists and Engineers*

Chapter Overview

This chapter provides an introduction to and "quick tour" of GoldSim. It presents the basic concepts of how simulation techniques can be applied to solve problems, and how a simulation model is created in GoldSim. It also provides an overview of the features and capabilities of the program.

If you read nothing else before starting to use the program, it is strongly recommended that you read this chapter, as it will tell you what the program is capable of doing, provide an overview of how to build a model, and direct you to those portions of the manual where you can obtain further information.

In this Chapter

This chapter discusses the following:

- Understanding Simulation
- What is GoldSim?
- Basic GoldSim Concepts
- Advanced Computational Features
- Documenting and Presenting Your Model
- Specialized GoldSim Modules
- The GoldSim Player
Understanding Simulation

Before describing what the GoldSim simulation framework is, and what it can be used to do, it is important to first discuss in general terms what we mean when we use the term simulation.

The term simulation is used in different ways by different people. As used here, simulation is defined as the process of creating a model (i.e., an abstract representation or facsimile) of an existing or proposed system (e.g., a business, a mine, a watershed, a forest, the organs in your body, the atmosphere) in order to identify and understand those factors which control the system and/or to predict (forecast) the future behavior of the system. Almost any system which can be quantitatively described using equations and/or rules can be simulated.

Simulations can be static or dynamic. In a static simulation, the system does not change with time. Static simulations can be useful for carrying out simple risk calculations and decision trees, in which there is no dynamic component to the model.

In a dynamic simulation, the system changes and evolves with time (in response to both external and internal influences), and your objective in modeling such a system is to understand the way in which it is likely to evolve, predict (forecast) the future behavior of the system, and determine what you can do to influence that future behavior.

In effect, a dynamic simulation is used to predict the way in which the system will evolve and respond to its surroundings, so that you can identify any necessary changes that will help make the system perform the way that you want it to. For example, a fisheries biologist could dynamically simulate the salmon population in a river in order to predict changes to the population, and quantitatively understand the impacts on the salmon of possible actions (e.g., fishing, loss of habitat) to ensure that they do not go extinct at some point in the future:

[Graph showing simulated number of spawning salmon over time]

Probabilistic Simulation

In many systems, the controlling parameters, processes and events may be uncertain and/or poorly understood. In a deterministic simulation, these parameters are represented using single values (which typically are described either as "the best guess" or "worst case" values). Probabilistic simulation is the process of explicitly representing this uncertainty by specifying inputs as probability distributions and specifying any random events that could affect the system.
If the inputs describing a system are uncertain, the prediction of future performance is necessarily uncertain. That is, the result of any analysis based on inputs represented by probability distributions is itself a probability distribution.

Hence, whereas the result of a deterministic simulation of an uncertain system is a *qualified statement* (“if we build the dam, the salmon population could go extinct”), the result of a probabilistic simulation of such a system is a *quantified probability* (“if we build the dam, there is a 20% chance that the salmon population will go extinct”). Such a result is typically much more useful to decision-makers who might utilize the simulation results.

A more detailed description of probabilistic simulation concepts, including a discussion of the advantages and disadvantages of probabilistic simulation, is provided in Appendix A, "Introduction to Probabilistic Simulation".

In order to illustrate what is involved in simulating a system, the steps necessary to carry out any kind of simulation are briefly summarized below:

1. **Define your objectives and measures of performance.** Before attempting to simulate a system, it is important to clearly identify what types of questions you are trying to answer with the model. The objectives of the model define the *performance measures* for the system. A performance measure is a model output by which you can judge the performance of the system (e.g., the salmon population in a river, or the net present value of a project).

2. **Develop the conceptual model.** The most important step in simulating any system is developing a *conceptual model* of the system. A conceptual model is a representation of the significant features, events and processes controlling the behavior of the system. It is essentially a body of ideas, based on available information, that summarizes the current understanding of the system.

3. **Create the mathematical model.** Once a conceptual model of the system is developed, it is necessary to represent it quantitatively within a *mathematical model*. A mathematical model consists of a set of input assumptions, equations and algorithms describing the system.

4. **Quantify the input parameters.** The mathematical model identifies specific inputs (e.g., the flow rate for a river, the financial discount rate) which are required in order to simulate the system. These must be quantified by specifying their values or probability distributions.

5. **Implement and solve the mathematical model using a computational tool.** After developing the mathematical model and quantifying all of the input parameters, the model must be implemented within a computational tool capable of solving the equations representing the system. This implementation of the mathematical model within a computational tool is referred to as the *simulation model*. For simple systems, the computational tool might be your brain (if you can solve the equations in your head), a calculator, or perhaps a spreadsheet program. For more complex systems, a specialized computer program (such as GoldSim) is required.

6. **Evaluate, explain and present the results.** The final step in the simulation process is to produce results, evaluate and draw conclusions from these results, and present them to interested parties.

For very simple simulations, the steps described above could be carried out in order, proceeding from step 1 to step 6. For most real-world simulations, however, these steps should be carried out in an iterative manner. That is, as
new data and/or new insights into the behavior of the system are obtained, the model is refined and reanalyzed.

Models which are constructed and continuously updated in such a manner can provide a systematic framework for organizing and evaluating the available information related to a complex system, and can act as management tools to aid in ongoing decision-making.

As will be discussed below, GoldSim was specifically designed to make the construction of such models as easy and intuitive as possible.

Simulation is a powerful and important tool because it provides a way in which alternative designs, plans and/or policies can be evaluated without having to experiment on a real system, which may be prohibitively costly, time-consuming, or simply impractical to do. That is, it allows you to ask “What if?” questions about a system without having to experiment on the actual system itself (and hence incur the costs of field tests, prototypes, etc.).

A few wide-ranging examples of how simulation can be used to ask such "What if?" questions in order to solve real-world problems in science, engineering and business are listed below:

**Wildlife management:** By simulating an animal population (such as salmon or elk), you can determine which combination of management practices is most likely to be successful at maintaining a stable population.

**Strategic planning:** By simulating the development, regulatory approval, and marketing of a new drug, you can determine a strategy that will maximize profits.

**Reliability and Systems Engineering:** By simulating the components, processes, failure modes and events controlling a complex engineering system (such as a space system, a machine, or a large industrial facility), you can predict the reliability of the system, and modify the design so as to increase reliability and decrease the probability and/or consequences of failures.

**Water resources:** By simulating the inflows and future demands on a water supply reservoir, you can optimize management practices to minimize the probability of needing to seek other sources and/or impose water use restrictions at some future date.

**Environment:** By simulating the performance of a proposed landfill, you can modify the design to minimize environmental impacts.

**Manufacturing:** By simulating the supply chain for a product, you can experiment with technological changes (e.g., improved communication) and changes in decision rules (e.g., ordering practices) to stabilize inventory levels throughout the chain.

**Resource Planning and Management:** By simulating a natural resource (such as a forest), you can determine which combination of management practices (e.g., selective harvesting, development of recreational facilities) maximizes the use of the resource and best satisfies the various stakeholders.

As will be shown in the following sections, GoldSim is flexible and powerful enough to be readily applied to any of these problems.
What is GoldSim?

GoldSim is a highly-graphical simulation program that runs on personal computers using the Windows operating system. Although it was specifically designed for carrying out dynamic, probabilistic simulations of complex systems, it can also be readily applied to simpler static and/or deterministic simulations.

Because simulation can be such a powerful tool for understanding and managing complex systems, a variety of graphical simulation tools currently exist. The following combination of features, however, makes the GoldSim approach unique:

- **GoldSim is user friendly, highly graphical, and very flexible.** It can be applied to nearly any kind of system. You start to build a model in an intuitive manner by literally drawing a picture (an influence diagram) of your system.

- **GoldSim was specifically designed to quantitatively address the inherent uncertainty which is present in real-world systems.** GoldSim provides powerful tools for representing uncertainty in processes, parameters and future events, and for evaluating such systems in a computationally efficient manner.

- **GoldSim provides powerful capabilities for superimposing the occurrence and consequences of discrete events onto continuously varying systems.** This allows for the realistic simulation of discrete events such as financial transactions, accidents, system failures, storms, labor strikes, and lawsuits.

- **GoldSim was designed to facilitate the construction of large, complex models.** You can build a model of your system in a hierarchical, modular manner, such that the model can readily evolve and add detail as more knowledge regarding the system is obtained. Other powerful features, such as the ability to manipulate arrays, the ability to “localize” parts of your model, and the ability to assign versions to a model which is constantly being modified and improved, further facilitate the construction and management of large models.

- **GoldSim is dimensionally-aware. GoldSim has an extensive internal database of units and conversion factors.** You can enter data and display results in any units. You can even define your own customized units. GoldSim ensures dimensional consistency in your models and carries out all of the unit conversions internally. As a result, when you use GoldSim, it is never necessary for you to carry out (error-prone) unit conversions.

- **GoldSim is highly extensible.** You can dynamically link external programs or spreadsheets directly into your GoldSim model. In addition, GoldSim was specifically designed to support the addition of customized modules (program extensions) to address specialized applications.

- **GoldSim allows you to create compelling presentations of your model.** A model that cannot be easily explained is a model that will not be used or believed. GoldSim was specifically designed to allow you to effectively document, explain and present your model. You can add graphics, explanatory text, notes and hyperlinks to your model, and organize it in a hierarchical manner such that it can be presented at an appropriate level of detail to multiple target audiences.
What is GoldSim?

These features allow GoldSim to be applied at multiple levels, depending on the nature of the application:

- as a powerful, flexible simulator;
- as a system integrator; and
- as a visual information management system.

A Powerful, Flexible Simulator

At the most fundamental level, GoldSim can be used as a powerful, flexible simulator. That is, you may only wish to apply it to a very specific problem in a technical discipline such as finance, industrial engineering, environmental science, or chemistry.

GoldSim’s graphical interface and powerful computational features facilitate a wide range of simulations, ranging from a simple homework assignment put together in less than an hour, to a complex professional application built over a period of several months.

A System Integration Tool

Most real-world problems are multi-disciplinary in nature. That is, the system being simulated actually consists of many subsystems, and the sub-models for each subsystem must typically be built by people from a wide variety of disciplines. For example, a model intended to help manage an ecological system (e.g., a river) in order to support management decisions to protect an endangered species (e.g., salmon) likely would include sub-models that are developed by biologists, urban planners, hydrologists, civil engineers, economists, forestry professionals, and social and political scientists (among others).

Unfortunately, in many such cases, the model builders get caught up in the details of their sub-models, and lose sight of the "big picture". The end result is typically separate sub-models which are unnecessarily complex. More importantly, the complex interactions and interdependencies between subsystems are often ignored or poorly represented. Such an approach not only wastes resources, but is often too complex to be explained (and hence used) effectively, and too poorly integrated to represent the entire system in a cohesive and realistic way.

What is needed for such complex, multi-disciplinary systems is a tool that can be used to integrate all of the various sub-models into a single total system model. A total system model focuses on creating a consistent framework in which all aspects of the system, as well as the complex interactions and interdependencies between subsystems, can be represented.

Because GoldSim is flexible and powerful enough to represent practically any aspect of your system, and because GoldSim provides unique capabilities for building your model in a hierarchical, modular manner, it is ideally suited to act as a system integrator. In fact, this was the original and primary objective around which GoldSim was designed.

A Visual Information Management System

Complex models often require a great detail of input data. These inputs may reside in databases, spreadsheets, or in written documentation. The user of a model (e.g., the author of the model, a reviewer of the model, or a decision-maker evaluating the results) can be most effective if this input information can be visually integrated with (and readily accessed and viewed alongside) the simulation model. In addition, for a complex model which requires a great deal of input, it may be desirable (or even mandated) that the simulation model can directly access various data sources (e.g., databases or spreadsheets) to ensure the quality of the data transfer. GoldSim was designed to facilitate both of these tasks.
Even if you can directly and visually access the input data for your model, in order for your simulation model to be useful, you must also be able to explain its assumptions (and the implications of the simulation results) in a compelling and effective manner. GoldSim provides the tools to enable you to do so.

Hence, at the highest and most powerful level, GoldSim can used as a visual information management system, providing you with the ability to directly link to data sources, as well as describe, document and explain your model in a compelling and effective manner to any audience.

**Basic GoldSim Concepts**

The GoldSim simulation environment is highly-graphical and completely object-oriented. That is, you create, document, and present models by creating and manipulating graphical objects representing the components of your system, data and relationships between the data:

![Diagram of GoldSim model components]

In a sense, GoldSim is like a "visual spreadsheet" allowing you to visually create and manipulate data and equations. As can be seen in the simple example shown above, based on how the various objects in your model are related, GoldSim automatically indicates their influences and interdependencies by visually connecting them in an appropriate manner. GoldSim also sets up and solves the equations represented by the objects and their interdependencies.

The various objects with which a GoldSim model is constructed are referred to as *elements*. Each element represents a building block of the model, and has a particular symbol or graphical image (which you can subsequently customize) by which it is represented on the screen. Typically, you give each element a unique name by which it is referenced (e.g., X, Revenue, Rainfall, Discount_Rate). Most elements accepts inputs, and in turn produce outputs.
GoldSim provides a wide variety of elements. Some of these elements provide a mechanism for the user to enter input data into the model. Other elements represent functions which operate on one or more inputs and produce one or more outputs. A special class of elements (stocks and delays) is critical for generating the dynamics in your models.

**Input Elements**

*Input elements* define the inputs to your model. There are five types of input elements: Data, Time Series, Stochastics, Lookup Tables and History Generators:

- **Data** elements allow you to specify a single scalar value (e.g., the discount rate) or an array of related values (e.g., the salaries of each individual in a group).
- **Time Series** elements allow you to specify a time series of a value (e.g., monthly rainfall rates, quarterly cashflows). Stochastic elements allow you to specify that a particular value is uncertain by defining it as a probability distribution.
- **Lookup Tables** and **History Generators** are somewhat more complex input elements. Lookup Tables can be used to define a response surface specifying how an output varies as a function of up to three inputs. History Generator elements generate random (stochastic) time series based on specified statistics (e.g., to represent the price of a security such as a stock or bond).

**Function Elements**

*Function elements* instantaneously compute outputs based on defined inputs. That is, they require one or more inputs, carry out a specified calculation on those inputs, and produce one or more outputs. GoldSim provides fifteen function elements.

The simplest and most general function element is an **Expression element**. Expressions produce a single output by calculating user-specified formulas, such as 2 + 3, A*B, or sqrt(55). Similar to a cell in a spreadsheet, when defining an expression, you can use a wide variety of mathematical operators and functions. You can even use conditional operators (e.g., >, <, =) and *if, then* logic to define Expressions.

Other function elements have predefined functionality. For example, a Selector element provides an easy and transparent way to specify if..then logic in a model.

**Stock and Delay Elements**

Stock and Delay Elements are specialized function elements with the unique property that their outputs are influenced by what has happened in the past. That is, unlike standard function elements, whose outputs at any given time are computed based solely on the current (instantaneous) values of their inputs, the outputs of these elements are determined by the *previous* values of their inputs. Such elements accumulate past events and provide systems with inertia and memory, and hence are responsible for internally generating the dynamic behavior of a system.

An example of a Stock element is the **Reservoir**. A Reservoir accumulates materials, and is useful for representing things like bank accounts and quantities of materials or items (e.g., water, soil, salmon). In its simplest form, a Reservoir
requires as inputs an Initial Value, a Rate of Addition and a Rate of Withdrawal, and outputs a Current Value using the following equation:

\[
\text{Current Value} = \text{Initial Value} + \int (\text{Rate of Addition} - \text{Rate of Withdrawal}) dt
\]

An example of a Delay element is the \textit{Material Delay}. Delays allow you to represent processes in which the output lags the input. The Material Delay accepts as input a flow of material (e.g., gal/day, $/yr, widgets/hr), moves it through a “conveyor” or “pipeline” (while optionally dispersing it), and then outputs the flow. Such an element can be used to represent such processes as the movement of water through soils or the movement of parts in a conveyor.

**Linking Elements**

GoldSim models are built by connecting the outputs of one (or more) elements to the inputs of other elements. These connections are referred to as links:

A complex GoldSim model can have hundreds (or thousands) of elements and links.

**A Simple Example**

The easiest way to understand GoldSim is to walk through a very simple example. In what follows, we illustrate how GoldSim could be used to carry out a simple dynamic simulation which computes the volume of water in a pond as a function of time.

Note: The objective here is not for you to follow along and try to build this model (as you have not yet learned the basics of how to do so). Rather, the objective is simply to provide a very general indication of how GoldSim models are constructed and the kinds of results they produce.

Assume that the pond that we wish to simulate is initially empty. We continuously pump water out of a nearby river and into the pond at a rate of 10,000 m\(^3\)/day. The pond leaks right back into the river, however, at a rate that is proportional to the current volume of water in the pond (i.e., the more water in the pond, the faster it leaks). In particular, 15\% of the pond volume is assumed to leak out every day. We wish to predict the water level in the pond as a function of time.

We will build this model using three elements: a Data element, a Reservoir and an Expression. The Data element represents the rate of inflow to the pond, the Expression calculates the leakage from the pond, and the Reservoir represents the volume of water in the pond.

Therefore, we begin by inserting a Data element named "Inflow", an Expression named "Leakage" and a Reservoir named "Volume_in_Pond":

```
\[
\text{Data element: Inflow}
\]
```

```
\[
\text{Expression: Leakage}
\]
```

```
\[
\text{Reservoir: Volume_in_Pond}
\]
```
Next, we define the properties of the Data element "Inflow" by double-clicking on the element, which causes its editing dialog to be displayed:

The name of the element is specified in the **Element ID** field. You add a brief description to the element in the **Description** field. The data defining the element is specified in **Data Definition** field. When writing the Data Definition, the units of the input are specified following the value.

To define the element, we enter the inflow rate of 10,000 m$^3$/day. Note that we specifically define not only the value, but also the dimensions (the units) of the input. Note also that we do not add punctuation (the comma).

Next, we define the Expression element representing the leakage (outflow) from the pond. The leakage is equal to 15% of the pond volume per day. Mathematically, this can be written as:

\[
\text{Leakage} = \text{Volume in Pond} \times 15 \text{%/day}
\]

We type the right-hand side of this equation into the input field for the Expression representing the leakage:
The expression representing the leakage is entered in the **Equation** field.

Finally, we must link the Inflow and Leakage to the Reservoir element. The properties dialog for a Reservoir (which is somewhat complex) is shown below.

It is assumed that the pond is initially empty. The Inflow and Leakage are specified as **Additions** and **Withdrawal Requests** from the Reservoir. The remaining input fields for the Reservoir are not utilized in this simple example.

For this example, the only required inputs for the Reservoir are the **Initial Value** (in this case, zero), the **Rate of Change Addition** (set equal to the Inflow), and the **Rate of Change Withdrawal Request** (set equal to the Leakage).

After defining the elements in this way, the screen looks like this:
The lines (referred to as "influence") between the elements will initially be drawn as straight lines (rather than as curve lines). GoldSim allows you to subsequently change the appearance of these influences (e.g., make them curved).

The arrows indicate how the three elements are related. In this case, the system has a feedback loop: Volume_in_Pond is a function of Leakage and Inflow, and the Leakage is a function of Volume_in_Pond. In fact, this is a classic example of a negative feedback loop: as Volume_in_Pond increases, the Leakage increases; as the Leakage increases, the Volume_in_Pond decreases. Negative feedback loops are self-correcting: rather than the pond volume continuing to increase, the feedback loop will eventually force it toward a steady state value.

GoldSim is a dynamic simulator, which means that your model can evolve and change with time. In this example, we are interested in predicting how the volume of water in the pond changes with time.

In order to do so, GoldSim must solve the equations represented by the elements and their linkages. In this particular example, the system can be represented by the following differential equation:

\[
\frac{\delta V}{\delta t} = \text{Inflow} - \text{Leakage} = \text{Inflow} - 0.15 * V
\]

where \( V \) is the volume in the pond, and \( \delta V/\delta t \) is the time derivative of the volume (the rate of change of volume with respect to time).

This particular equation can actually be solved analytically:

\[
V(t) = V(0) * e^{-0.15t} + \frac{\text{Inflow}}{0.15} \left[ 1 - e^{-0.15t} \right]
\]

In general, however, a particular system will not have an analytical solution (i.e., it cannot be solved exactly), and must be solved numerically (i.e., using an algorithm that provides a numerical approximation to the actual solution).

GoldSim solves differential equations such as this through numerical integration. That is, GoldSim solves for \( V \) as a function of time by integrating the right-hand side of the equation:

\[
V(t) = V(0) + \int_0^t \left( \text{Inflow} - 0.15 V \right) dt
\]

To solve this (or any) integral numerically, it is necessary to discretize time into discrete intervals referred to as timesteps. GoldSim then “steps through time” by carrying out calculations every timestep, with the values at the current timestep computed as a function of the values at the previous timestep. In the case of the
integral represented above, after discretizing time, GoldSim solves the equation by numerically approximating it as a sum:

\[ V(\tau) = V(0) + \sum_{0}^{t} \left[ \text{Inflow} - 0.15 \times V(\tau - \Delta t) \right] \Delta t \]

where \( \Delta t \) is the length of the timestep.

Hence, the value of \( V \) at time \( \tau \) is computed based on the value of the Leakage (which is in turn a function of \( V \)) at the previous timestep. This particular numerical integration method (referred to as \textit{Euler integration}) is discussed further in Appendix G.

In order to dynamically simulate a system in GoldSim, you must specify the duration of the simulation (e.g., 1 year) and the number of timesteps to use (e.g., 365). This information determines the length of the timestep (e.g., 1 day). The appropriate timestep length is a function of how rapidly the system represented by your model is changing: the more rapidly it is changing, the shorter the timestep required to accurately model the system.

For this simple example, we choose to run the model for a duration of 50 days using 50 timesteps (resulting in a timestep length of 1 day). The result, in the form of the volume of water in the pond as a function of time, is shown below:

As can be seen, the pond reaches a constant (steady-state) value after about 40 days. At this point, the flow rate into the pond matches the leakage rate out of the pond.

As you no doubt noticed in the example, when data were entered, the dimensions were specified. One of the more unique and powerful computational features of GoldSim is that the program is \textit{dimensionally aware}.

GoldSim has an extensive internal database of units and conversion factors. You can enter data and display results in any units. You can even define your own customized units.

When elements are created, you must specify their output dimensions. For example, when the Inflow element was created in the previous example, it was defined to have dimensions of volume per time (m\(^3\)/day). When elements are linked, GoldSim ensures dimensional consistency and carries out all of the unit conversions internally. For example, you could add meters and feet in an equation, and GoldSim would internally carry out the conversion. Note, however, that if you tried to add meters and hours, GoldSim would issue a warning message and prevent you from doing so.
As a result, when you use GoldSim, it is never necessary for you to carry out (error-prone) unit conversions.

*Read more:* [Using Dimensions and Units](#) (page 89).

In many systems, you may be uncertain about some of the input parameters. In such a case, GoldSim allows you to define these parameters as probability distributions.

In the example presented above, suppose that we were uncertain of the leakage fraction. In particular, we will assume that rather than being 15%, the leakage fraction could be anywhere between 10% and 20% of the pond volume per day.

To represent this, we create a new *Stochastic* element (named Fraction), and modify the equation for Leakage as follows:

\[
\text{Leakage} = \text{Volume\textunderscore in\textunderscore Pond} \times \text{Fraction}
\]

A Stochastic element is used to define a probability distribution. In this example, we will define Fraction (the leakage fraction) as a triangular distribution, with a minimum value of 10%/day, a most likely value of 15%/day, and a maximum value of 20%/day:

The type of probability distribution is selected from the list at the top of the dialog box. The parameters describing the shape of the distribution are defined under “Parameters”.

The model would then look like this:
If the inputs to a model are uncertain, the outputs are necessarily uncertain. GoldSim propagates (translates) the input uncertainties into uncertainties in the results using Monte Carlo simulation. In Monte Carlo simulation, the entire system is simulated a large number (e.g., 1000) of times. Each simulation is assumed to be equally likely, and is referred to as a realization of the system. For each realization, all of the uncertain parameters are sampled (i.e., a single random value is selected from the specified distribution describing each parameter). The system is then simulated through time (given the particular set of input parameters) such that the performance of the system can be computed.

This results in a large number of separate and independent results, each representing a possible “future” for the system (i.e., one possible path the system may follow through time). For example, 25 realizations of the pond example using an uncertain Leakage fraction produces 25 separate time history results:

Another way to view these results is by plotting a probability of the final volume of water in the pond (after 50 days):

This plot (presented as a cumulative distribution function) indicates that the median final volume is about 65,000 m$^3$, and there is a probability that the final volume could be as high as about 95,000 m$^3$.

Probabilistic simulations are particularly useful because you can often use them to obtain a better understanding of the factors controlling the system.

For example, by plotting the final volume in the pond as a function of the leakage fraction, we can better understand and quantify the sensitivity of the system to this key factor:
A primer on probabilistic simulation is presented in Appendix A.

Simple models have a direct chain of causality: input data affect some elements, which affect other elements, and so on, until eventually the elements which calculate the desired results of the model are reached:

GoldSim automatically analyzes your entire model to identify "who affects who", and ensures that the "upstream" elements are calculated prior to the "downstream" elements. At each time-step the elements are updated in this causality-based sequence.

Many systems, however, contain elements whose output can, directly or indirectly, affect one of their own inputs. This creates a looping or circular system (i.e., a "feedback loop").

Feedback loops are present in one form or another in most real-world systems, and can be readily modeled in GoldSim. The simple pond model presented previously is a good example of a system with a feedback loop. In the example above, A affects B which affects C, and then C “feeds back” to affect A and so on. That is, feedback loops represent a closed chain of cause and effect. Note that the terms “feedback” and “cause and effect” intentionally imply that the relationship between the variables is dynamic and the system changes over time (although systems with feedback loops can also reach a dynamic equilibrium).

In some cases, however, you may need to define circular logic in your model that is not dynamic at all. A simple simultaneous system of equations is an example of such a system. Such logic does not represent a feedback loop, in that there is no chain of “cause and effect” and no dynamics – in fact, the variables may not change with time at all. In GoldSim, these are referred to as recursive loops, and they are treated differently from feedback loops. GoldSim can still solve such systems, but in order to do so, it is necessary for you to take some additional steps to define the sequence in which the calculations are carried out.
Simulating Delays

In many systems, a signal (e.g., a flow of material, a piece of information) is not transmitted instantaneously within a system or process, but instead is delayed. Examples include the movement of water through soils, the movement of material along a conveyor, and the transfer of information from one person to another.

Material or information can be conceptualized as moving through (transiting) a delay process. In some cases, the material or signal may be dispersed while in transit. For example, if you post 100 letters all at once, they will not be delivered all at once. Rather, there will be some variability in the time at which they are delivered (i.e., the delay time). In other cases, the material or signal is not dispersed. If a conveyor belt moves at a fixed speed, there will be no variability in the transit times for items that are placed on the conveyor.

Delays can have a major impact on the dynamics of a system, and it is therefore important to be able to represent them in a straightforward and realistic manner. GoldSim provides several types of powerful Delay elements to represent these delays.

Read more: Using Delay Elements (page 320).

Building Hierarchical Top-Down Models

Complex models can have hundreds (or thousands) of elements. In order to manage, organize and view such a model it is useful (in fact, essential) to group the elements into subsystems. A subsystem is simply a collection of elements.

Subsystems are created in GoldSim by placing elements into Containers. A Container is simply a “box” into which other elements have been placed. In a sense, it is like a directory folder on your computer.

Containers can be placed inside other Containers, and any number of levels of containment can be created. This ability to organize model elements into a hierarchy provides a powerful tool for creating “top-down” models, in which the level of detail increases as you “push down” into the containment hierarchy.

The example below shows a system that has been divided into a number of distinct subsystems:
GoldSim provides an optional browser view of the system in a pane on the left of the window. The tree in the browser shows the containment hierarchy in a manner similar to how a computer's directory hierarchy is shown.

You can "drill down" into the next level of detail in the model by "opening" the Container. (One way to do this is by clicking on small triangle in the upper left-hand side of the Container.)

The ability to create hierarchical, top-down models, coupled with GoldSim's powerful documentation and presentation abilities, allows you to effectively describe and explain your model at different (and appropriate) levels of detail to different audiences.

Read more: Understanding Containers (page 95).

Although you can create complex models using only the elements introduced in the previous sections, GoldSim provides a variety of additional elements that provide further modeling capabilities, including:

- **Logical elements**, which allow you to add easy-to-read logic diagrams to your models;
- A **Selector**, which provides a powerful and user-friendly way to create complex equations involving nested if...then logic; and
- An **Extrema**, which dynamically computes the highest or lowest values achieved by a specified parameter throughout a simulation (e.g., peak concentration in a river, minimum water level in a reservoir, minimum balance in a checking account).

Read more: Chapter 4: Using the GoldSim Elements (page 123).

Advanced Features

In addition to the basic features and capabilities discussed above, GoldSim provides a large number of advanced computational features which increase the power, flexibility and ease-of-use of the software. Several of these features are discussed below.
In many systems, you will want to create and manipulate elements that represent collections of data, rather than individual items. For example, you may want to create an element that represents your company’s revenue for each of five separate divisions, or an element that represents the salmon population in each of a number streams.

One way to do this, of course, would be to create separate elements for each object you wanted to model (e.g., five elements representing revenue, 13 elements representing salmon populations in different streams). Such an approach, however, is not desirable for two reasons:

- It could require you to create a very large number of elements (e.g., if you wanted revenues for 100 subgroups or wanted to evaluate salmon populations in 200 streams). This could result in very large, cluttered models.
- Usually, you will want to carry out the same types of calculations and operations on all related objects in such a collection (e.g., multiply all revenues by 2, compute the number of salmon eggs this year for all the streams based on the current salmon population). Having to do this individually for every object in a large collection would be very cumbersome and time-consuming.

To address these kinds of problems, GoldSim allows you to create and manipulate vectors and matrices (collectively referred to as arrays). For example, you could create a vector Data element that represented the salmon populations in each of the streams, as shown below:

You could also create a matrix Data element that represented the salmon population in each of a number of streams for a period of 3 years, as shown below:
In addition to adding data in the form of vectors and matrices, you can manipulate these arrays in equations. For example, you could create an Expression element, and define it as:

\[ 2 \times \text{Salmon\_Population} \]

The output of the Expression would be an array, identical to the \text{Salmon\_Population} array, except each item of the array would be two times greater. Note that if you did not use arrays to carry out this calculation, rather than creating 2 elements, you would need to create 26 elements (2 for each stream: 13 Data elements and 13 Expression elements) to accomplish the same thing!

If required, you can access a particular item of the array in an equation. For example,

\[ \text{Salmon\_Matrix[BearCreek, 1999]} \]

references a single (scalar) value representing the salmon population in Bear Creek in 1999.

GoldSim provides a wide variety of special operators which allow you to manipulate arrays. For example, if the name of the vector was \text{Salmon\_Population}, the expression

\[ \text{Sumv(Salmon\_Population)} \]

would result in a single (scalar) value representing the sum of all items in the vector.

Read more: Using Vectors and Matrices (page 782).

GoldSim provides powerful capabilities for superimposing the occurrence and effects of discrete events onto continuously varying systems. This allows for the realistic simulation of discrete events such as financial transactions, accidents, system failures, storms, labor strikes, and lawsuits. Events such as these can have important effects on the performance of many systems, and it is therefore important to represent them in a realistic manner.

GoldSim provides a variety of specialized elements for simulating the occurrence and consequences of discrete events. Events can be generated regularly ("occur exactly once a year on January 1"), randomly ("occur, on average, once a year"), or based on certain set of conditions ("when A is greater than B and the value of C has changed"). An event can trigger one or more consequences, such as changing the status of something in the model ("this task
is now complete”), achieving a specified milestone, or making a discrete change to some quantity in your model (“add $1000 to the account”).

To provide an indication of how you can use this feature, let’s add a simple event to the example involving the pond presented above in “A Simple Example”. Suppose that water in the pond is used for fire suppression, and if a brush fire occurs nearby, firefighters withdraw 10,000 m$^3$ from the pond. Let’s assume that it is summer, and there is a small fire nearby about once every 25 days.

We could simulate this using GoldSim’s Timed Event and Discrete Change elements:

The Event “Fire” triggers the Discrete Change “Withdrawal” which modifies the Reservoir element representing the volume in the pond.

The result of such a simulation would look like this:

In this particular realization of the system, fires occurred around the 10$^{th}$ and 32$^{nd}$ days of the simulation. Because the event was specified as being random, it did not occur exactly every 25 days.

Read more: Chapter 5: Simulating Discrete Events (page 351).

GoldSim provides the ability to make subsystems of your model (i.e., Containers) conditional. Conditionality allows you to make a Container and all of its contents inactive unless specific events occur and/or conditions are met. Elements in an inactive container are “dormant”. That is, they are not updated or recalculated each timestep, and while they are inactive their output values never change. When other specific events occur and/or conditions are met, the Container (and its contents) can become active (and hence carry out their normal calculations). A conditional Container can activate and deactivate multiple times during a simulation.
Conditionality is a very powerful feature, and can be used to 1) temporarily “turn off” certain parts of your model (e.g., during a testing phase); or 2) simulate processes or features which themselves only exist or are active during certain parts of your simulation. This feature is particularly useful when using GoldSim to simulate projects.

Read more: Using Conditional Containers (page 899).

GoldSim provides a powerful timestepping algorithm that allows you represent the dynamics of your system very accurately. This includes the following:

- You can define Reporting Periods, which compute and report accumulated, average, the change or the rate of change of values over specified periods (e.g., monthly, annually). Reporting Periods can be used in conjunction with a shorter “Basic Step” (e.g., you may model the movement of water or money through a system using a daily timestep, but need to report the cumulative amount of money or water that moved from one point to another each month).

- You can increase or decrease the timestep length according to a specified schedule during a simulation (e.g., start with a small timestep, and then telescope out to a larger timestep). This can be useful, for example, if you know that early in a simulation, parameters are changing rapidly, and hence you need a smaller timestep.

- You can dynamically adjust (adapt) the timestep during a simulation based on the values of specified parameters in your model. For example, you could instruct GoldSim to use a timestep of 1 day if X was greater than Y, and 10 days if X was less than or equal to Y. Similarly, you could instruct GoldSim to use a short timestep for a period of 10 days after a particular event occurs, and then return to the default timestep.

- You can apply dynamic adaptive timestepping to specific Containers. This allows you, for example, to specify different timesteps for different parts (i.e., Containers) in your model. For example, if one part of your model represented dynamics that changed very rapidly (requiring a 1 day timestep), while the rest of the model represented dynamics that changed much more slowly (requiring a 10 day timestep), you could assign a 10 day timestep to the model globally, and a 1 day timestep to the Container representing the subsystem that changed rapidly.

- For some special types of systems, GoldSim provides additional dynamic timestepping algorithms (different from the timestep algorithms described above) to more accurately solve these equations. In particular, the Contaminant Transport Module utilizes dynamic timestep adjustment to accurately solve the coupled differential equations associated with mass transport.

Read more: Defining Reporting Periods (page 461); Advanced Timestep Options (page 466).

In some models, you may want to carry out an iterative calculation at each timestep. This might be useful, for example, if you have a coupled system of nonlinear equations that must be solved every timestep by iterating.

GoldSim allows you to create a Looping Container in which the elements inside are iteratively updated every timestep until a specified condition is met:
Such a looping calculation is facilitated by the fact that GoldSim also provides a specialized element that allows you to reference the previous value (of any variable in your model).

Read more: Using Looping Containers (page 963); Referencing an Output’s Previous Value (page 957).

Dynamically Linking to Spreadsheets

GoldSim allows you to dynamically link a spreadsheet directly into your model. In the simplest use of such a spreadsheet link, you can use the spreadsheet as a data source. In particular, a Spreadsheet element can import data from specified cells in a spreadsheet, assign specified units to these data, and make them available in your GoldSim model (as outputs of the element).

Note: When you import a row or column of spreadsheet cells into GoldSim (e.g., A1:A10), the data become items in a vector within GoldSim. When you import a range of cells involving multiple columns and rows (e.g., A1:E10), the data become items in a matrix within GoldSim.

In addition to importing data from a spreadsheet, you can use a spreadsheet as a custom element (with specific functionality). That is, you can dynamically send data from GoldSim to a spreadsheet, force the spreadsheet to recalculate, and then retrieve (updated) data from the spreadsheet back into GoldSim during a simulation (e.g., every timestep).

GoldSim also provides powerful capabilities to easily import lookup tables and time series data from spreadsheets to GoldSim. You can also readily export GoldSim results to a spreadsheet.

Read more: Spreadsheet Elements (page 910); Linking a Lookup Table to a Spreadsheet (page 306); Importing Data into a Time Series from a Spreadsheet (page 194); Exporting from a Time History Result Element to a Spreadsheet (page 720).
Importing Inputs from a Database

In simulations which require a great deal of input, it may be desirable (or even mandated) that the simulation model can access the various data sources directly to facilitate and ensure the quality of the data transfer.

As discussed above, one way to accomplish this in GoldSim is to import data from spreadsheets. GoldSim also provides a more powerful method. In particular, GoldSim input elements can be linked directly to an ODBC-compliant database. After defining the linkage, you can then instruct GoldSim to download the data at any time. When it does this, GoldSim *internally records the time and date at which the download occurred*, and this information is stored with the model results. This mechanism provides very strong and defensible quality control over your model input data.

*Read more:* [Linking Elements to a Database](page 1031).

Building Custom Elements Using Scripts

In some situations, you may wish to define a complex function which cannot be readily implemented using the expression editing features supplied by GoldSim. For example, calculation of an output may require very complex logic which would be cumbersome to represent using a Selector element, or it may require a numerical solution technique (e.g., iteration); or perhaps you need to construct an array using complex logic.

To deal with such situations, you can specify a sequence of statements directly within the GoldSim interface using a Script element; in effect, building a custom element.

Scripts are created by inserting and editing statements or statement blocks, which may be variable definition statements, variable assignment statements, statements controlling the sequence of execution in the script (e.g., loops and if statements), or statements used for writing messages to the Run Log. The Script element sequentially evaluates the specified sequence of locally defined statements to determine its output(s).

*Note:* The Script element does not expect the user to learn or be familiar with a particular language. As a result, scripts are not created using a text editor. Rather, statements are inserted and edited within a “controlled environment” within the element’s property dialog in which the user selects from a number of available statement types. The syntax is already defined for each type of statement – the user simply specifies the attributes and properties for each statement via a dialog box when the statement is inserted. Statements can subsequently be moved, deleted, and edited.

*Read more:* [Script Elements](page 860).

Dynamically Linking to External Models

If GoldSim’s built-in elements, the Script element or a dynamically-linked spreadsheet are not sufficient for representing a particular aspect of your model, you can dynamically link an external computer program directly to GoldSim. You must specify the parameters (outputs of existing GoldSim elements) which you wish to send to the external program, and the parameters that the external program will return to, and make available within, GoldSim.

In most cases, you need to only make minor modifications to the external program code to which you want to link in order for it to communicate with (i.e., be dynamically called by) GoldSim.

Note that this ability greatly increases the power and flexibility of GoldSim, allowing nearly any program to be dynamically linked into the dynamic, probabilistic and highly graphical GoldSim framework.
Building Large, Complex Models

In addition to the ability to build hierarchical models using Containers, GoldSim also provides a number of other features that were specifically designed to facilitate the construction, maintenance, and presentation of very large, realistic (and hence, often complex) models.

This includes:

- The ability to “localize” subsystems in your model so that variable names can be repeated without causing conflicts. This is particularly useful when your model contains numerous parallel systems (e.g., different divisions of a company), in which many of the equations and variable names would be identical. It also allows multiple people to work on different subsystems in a model without worrying about conflicting variable names.

- The ability to record versions (revisions) of a particular model file, so that you can identify the differences between the various versions of the file as the model is iteratively modified. (Which elements have changed? Which elements were deleted? Which elements have been added?)

Optimizing a Model

GoldSim provides the ability to carry out a special type of run to facilitate optimization of your model. For this type of run, you specify an objective.
function (a specific result that you would like to minimize or maximize), an optional constraint (a condition that must be met), and one or more optimization variables (variables in your model that you have control over).

GoldSim then runs the model multiple times, systematically selecting combinations of values for each of the optimization variables. By doing so, GoldSim can determine the values of the optimization variables that optimize (minimize or maximize) the objective function while meeting the specified constraint.

Typical uses of optimization include:

- Finding the best input data values for a model, in order to match observed historical data (i.e., calibration).
- Selecting the “best” option from among alternatives. “Best” could mean safest, cheapest, most reliable, or another appropriate measure.
- Optimizing the timing of actions or policy changes during the course of a simulation.

Read more: Running an Optimization (page 528).

GoldSim provides the ability to carry out a special type of run to facilitate sensitivity analyses. For this type of run, you specify the result you are interested in, and one or more variables that you want to analyze (which must be Stochastics or Data elements).

GoldSim then runs the model multiple times, systematically sampling each variable over a specified range, while holding all of the other variables constant. This then allows GoldSim to produce sensitivity plots (i.e., a tornado chart and X-Y function charts) to assist you in graphically identifying the variables in your model to which the result is most sensitive.
GoldSim also provides a second type of sensitivity analysis in which statistical sensitivity measures are computed by analyzing the results of multiple realizations of the model where all of the Stochastic variables are simultaneously sampled each realization:

Read more: Running Sensitivity Analyses (page 539); Viewing a Sensitivity Analysis Table (page 681).

Features for Documenting and Presenting Your Model

A model which cannot be easily explained is a model that will not be used or believed. As a result, GoldSim was specifically designed to allow you to effectively document, explain and present your model.

You can add graphics, explanatory text, notes and hyperlinks to your model, and organize it in a hierarchical manner such that it can be presented at an appropriate level of detail to multiple target audiences.

GoldSim has powerful charting and display functions that allow you to plot and view your data in a variety of ways. You can plot time histories of your data, view probability distributions, create scatter plots and bar charts, and view tables of results. You can also combine multiple results on a single plot, and view multiple plots simultaneously. Using Result elements, result charts and tables can be easily collected into one place for easy access within a model.

In addition, you can modify and save chart styles, which allow you to customize (and reuse) the style (i.e., appearance) for each type of chart you may wish to produce:
Using these tabs, you have complete control over the appearance of your chart. You can customize the header, footer and legend, modify the chart axes, and change the data style (e.g., color, size and type of line). You can create also chart styles, which can subsequently be applied to similar charts.

The charts and tables produced by GoldSim can be pasted into other applications (e.g., a word processor or a spreadsheet), or saved as separate graphics files.

Read more: Chapter 8: Displaying Results in GoldSim (page 555).

In order for a complex model to be understood and easily explained to others, it is critical that it be properly documented. GoldSim provides tools that allow you to internally document your model such that the documentation becomes part of the model itself, and hence is immediately available to anyone viewing the model.

GoldSim allows you to add text, images and other graphic objects directly to your model. In addition, you can add hyperlinks to other documents (e.g., a web site or a report). Clicking on the hyperlink then opens that document.
You can add text in any font or color, such as the title and description shown here. GoldSim provides drawing tools you can use to add text, lines, boxes (such as the one surrounding the four elements on the left) and other graphical objects. Hyperlink objects, such as the one labeled "Learn More", can link directly to a separate document or a web site. You can also import graphics, such as logos (as seen here) or maps.

Note: To further facilitate documentation and presentation of your model, the default graphic images for a GoldSim element can be replaced by a custom image of your choice. For example, if a Reservoir element was being used to represent the population of a city, you could replace the default Reservoir image with an image of people or buildings.

Adding Descriptions and Notes

In addition to adding graphics, text and hyperlinks, you can provide a Description for each element. The Description is displayed in a tool-tip whenever you place your cursor over the element in GoldSim.

To provide more information, you can also attach a Note to any element in your model. Notes are text files of arbitrary length that are stored with the model:
Using GoldSim as a Presentation Tool

As discussed previously, GoldSim was specifically designed to allow you to organize model elements into a hierarchy (using containers). This facilitates the creation of "top-down" models, in which the level of detail increases as you "push down" into the containment hierarchy.

Such a capability is essential if you wish to effectively describe and explain your model at different levels of detail to different audiences. For example, your manager may only want to see the "big picture", while a technical colleague may want to see the low-level details of a particular model.

The ability to create hierarchical, top-down models, in which at any level, details can be "hidden" (inside containers), coupled with GoldSim's powerful documentation features, allows you to design models which can be effectively explained to any audience at the appropriate level of detail.

Read more: Chapter 9: Documenting and Presenting Your Model (page 737).

Specialized GoldSim Modules

Although the standard elements incorporated within GoldSim can be used to build powerful and complex models, it was realized from the outset of the development of GoldSim that in some situations specialized elements and features may be required in order to efficiently model some kinds of systems. As a result, GoldSim was designed to readily facilitate the incorporation of additional modules (program extensions) to enable the program to address specialized problems.

Because extension modules can be highly specialized, they are described in separate documents.

The existing GoldSim extensions are briefly described below.

Financial Module
The Financial Module allows you to probabilistically simulate financial systems that include components such as accounts and funds, investments, options, projects or undertakings with specified cash flows, and insurance policies:

![Financial Module elements](image)

*The Financial Module adds these five specialized elements to GoldSim.*

By combining the specialized financial elements in the Financial Module with GoldSim's underlying probabilistic, dynamic simulation framework, you can quickly simulate and analyze complex financial systems, as well as complex engineering and business systems that have a financial component.

The Financial Module is described in detail in the **GoldSim Financial Module User's Guide**.

The Contaminant Transport Module consists of specialized elements for representing contaminant species, transport media, transport pathways, contaminant sources, and receptors, and the coupled sets of differential equations underlying these systems:

![Contaminant Transport Module elements](image)

*The Contaminant Transport Module (RT version) adds these twelve specialized elements to GoldSim.*

**Note:** Two versions of the Contaminant Transport Module are available (called CT and RT). RT is a premium version of the Contaminant Transport Module, and contains some specialized features and elements not included in the CT version.

By linking these environmental elements together (and integrating them with GoldSim's basic elements), you can build complex contaminant transport simulations.

The Contaminant Transport Module is described in detail in the **GoldSim Contaminant Transport Module User's Guide**.

The GoldSim Reliability Module allows you to probabilistically simulate the reliability of complex engineered systems over time. GoldSim provides the ability to model the interdependence of components through requirements and fault trees, as well as the capability to define multiple independent failure modes for each component.

**Reliability Module**

GoldSim User's Guide  Chapter 2: GoldSim in a Nutshell  ●  53
The fundamental outputs produced by the Reliability Module consist of predicted reliability metrics (e.g., reliability, maintainability and availability) for the overall system, and for individual components within that system. In addition, GoldSim catalogs failure scenarios, which allows for key sources of unreliability to be identified.

The Reliability Module can be used to:

- evaluate the reliability performance of systems and ascertain their compliance with customer or regulatory requirements;
- investigate the impact of the operational environment on system reliability; and
- simulate the effects of reliability on system throughput, or on other user-defined metrics.

The Reliability Module is described in detail in the GoldSim Reliability Module User’s Guide.

Dashboard Authoring Module

GoldSim provides a set of tools that allow a GoldSim modeler to design and construct “dashboard” interfaces for models. The interfaces can be designed to look like “dashboards” or “control panels”, with buttons, gauges, sliders and display panels, and the author can imbed instructions, tool-tips and graphics to provide instructions on the use of the model. Such an interface allows a model to be easily used by someone without requiring them to be familiar with either the GoldSim modeling environment or the details of the specific model:
In effect, the Dashboard Authoring tools enable you to use GoldSim as a high-level programming language to create custom simulation applications for distribution to end users who may not necessarily be modelers. Users do not need to have a licensed version of GoldSim to view and run a "dashboarded" model. They simply need to download the free GoldSim Player.

Read more: The GoldSim Player (page 55).

The Dashboard Authoring tools are incorporated into all versions of GoldSim. They are described in detail in a separate document, the GoldSim Dashboard Authoring Module Users Guide.

Monte Carlo simulation, in which multiple realizations of a system are simulated, naturally lends itself to distributed processing, since each realization is totally independent of the others. GoldSim has the capability to utilize multiple computers (linked over a network such as a LAN or even the Internet) to carry out probabilistic simulations of a system. Such a feature greatly facilitates probabilistic simulation of highly complex systems (in which a single realization may take many minutes or even hours).

The Distributed Processing Module is described in detail in the GoldSim Distributed Processing Module Users Guide.

The GoldSim Player

The GoldSim Player is a special version of GoldSim that allows you to "play" or "read" an existing GoldSim model without having to license the GoldSim software.

In general, the user interface for the GoldSim Player is identical to that of the full GoldSim version, with menu options and controls for editing the model removed or disabled. You can view and navigate any GoldSim model using the GoldSim Player. This allows a modeler to distribute a model to others without requiring them to license GoldSim.
If a Dashboard was created for the model using the Dashboard Authoring Module, you can even modify selected inputs and run the model using the GoldSim Player.

The GoldSim Player can be downloaded (for free) from the GoldSim Web site (www.goldsim.com/player).

*Read more:* Dashboard Authoring Module (page 54); Creating a GoldSim Player File (page 777).
Chapter 3: Building a Model in GoldSim

The great successful men of the world have used their imaginations, they think ahead and create their mental picture, and then go to work materializing that picture in all its details, filling in here, adding a little there, altering this a bit and that a bit, but steadily building, steadily building.

Robert Collier

Chapter Overview

This chapter describes the GoldSim user interface and presents the basic techniques required to build a model. The objective of the chapter is to acquaint you with the user interface components and provide you with the procedural knowledge required to navigate and carry out basic functions in the user interface. Conceptual issues (e.g., which element should I use to accomplish a particular task?) are discussed in the following chapter (Chapter 4).

Since the only way to learn a new tool is to use it (rather than just read about it), it is highly recommended that as you read this chapter, you simultaneously begin to use and explore the program. That is, as a feature or action is described in the text, experiment with it yourself within GoldSim.

Warning: This chapter assumes that you have read the previous chapter, "GoldSim in a Nutshell", which provides some basic background information on the features and capabilities of GoldSim.

In this Chapter

The following topics are discussed in this chapter:

- The GoldSim User Interface
- Creating Elements and Links
- Navigating and Viewing a Model
- Running a Model and Viewing Results
The GoldSim User Interface

Whenever you start GoldSim, the following dialog is displayed:

The GoldSim Start Dialog

The choices on the Start dialog are as follows:

- **New Model**: This opens a new (blank) model.
- **Open Model**: This displays a dialog for opening an existing model.
- **Recent Models**: The eight most recently opened models are displayed here. Holding your cursor over a filename displays information (path, date modified, and size) regarding the file. Clicking on the filename opens the model.

You can also close the Start dialog and open a new model (without displaying the Simulation Settings dialog) by clicking anywhere outside of the dialog or pressing the *Esc* key.

Pressing **Help Center** will take you to the GoldSim Help Center, the primary web portal for accessing technical support. Pressing **Tutorial** will open a Tutorial. The GoldSim Tutorial presents the basic concepts on which GoldSim is based and provides an overview of GoldSim’s key features and capabilities.

*Read more:* [Getting Technical Support](#) (page 21).

User Interface Components

When you open a new model in GoldSim, you see the following screen:
The largest part of the screen is occupied by the **graphics pane**. This is where the graphical depiction of the model is shown. It is the "canvas" on which the model is drawn. For most users, this is where most model building and manipulation tasks will be carried out.

---

**Note:** If you have activated a GoldSim extension module, there may be some default items present in your models. For example, the Contaminant Transport Module adds a Container named Materials.

---

When you open GoldSim for the first time, the browser will be shown. The browser represents an alternative way to view a GoldSim model. In particular, it organizes the model in one of two ways:

- In a hierarchical manner (by containment), similar to the way that files and directories on a computer are organized;
- By element type.

The browser is initially docked on the left side of the screen. You can move it, undock it (so that it floats as a separate window) or deactivate (hide) it. You can turn the browser off and on by pressing the browser toolbar button or by pressing F6.

**Read more:** [Using the Browser](page 105).

The **menu bar** provides access to menus from which nearly any GoldSim operation can be carried out. Although there are sometimes more convenient ways to carry out a particular action in GoldSim (e.g., toolbar buttons, context-sensitive menus), if you are stuck, you will find that most GoldSim functions and operations can be carried out via the main menu.

---

The user interface also includes a number of toolbars. All toolbars are present the first time you open GoldSim. All toolbars (as well as the menu bar) can be customized, hidden, and moved if desired. (In the screenshot above, several of the toolbars have been hidden.) The two most commonly used toolbars are the **Standard toolbar** and the **Run Control toolbar**.

- The Standard toolbar provides buttons for common GoldSim actions (e.g., Save a file, Run the model):

![Standard toolbar buttons](image)

- The Run Control toolbar provides buttons for running (and pausing) a model:

![Run Control toolbar buttons](image)

The **status bar** at the bottom of the screen provides information regarding the mode that the model is in, as well as access to tools (e.g., zooming). It also indicates the status of a simulation while it is in progress.

The appearance of the user interface components can be configured and customized to match the way that you like to use GoldSim. For example, you can customize toolbars, change the appearance of the graphics pane, and modify the appearance of objects in the interface.

**Read more:** [Chapter 6: Customizing the Interface](page 421).
The GoldSim simulation environment is highly-graphical and completely object-oriented. That is, you build, document, and present models by creating and manipulating objects representing variables and relationships between the variables. For the purposes of discussing how to manipulate objects in GoldSim, it is important to differentiate between two types of objects: model objects and graphical objects.

A model object is used to quantitatively represent the variables and relationships in your model. Elements, the basic building blocks of a GoldSim model, are the primary model objects. Input and Output objects associated with each element, as well as Influences, are also model objects.

Read more: Element Inputs and Outputs (page 74); Understanding Influences (page 98).

A graphical object is used to embellish or document the model. Graphical objects consist of images, text, lines, and other graphics that you can add to your model in order to document it. They have no impact on the simulation model itself, and only serve to improve the way in which the model is presented and documented. This chapter focuses on the use of model objects.

There are some differences in the manner in which these two types of objects are manipulated in GoldSim, and this is specifically noted throughout the documentation. In particular, whenever the word "object" is used to describe an action or behavior, it applies to both model objects and graphical objects. If a particular action or behavior only applies to a certain type of object, the object type (e.g., model object, element, or graphical object) is explicitly used.

The most common and important GoldSim mouse actions are summarized below:

A click selects an object in the graphics pane or the browser. The selected object is highlighted.

Note how the selected element (Expression1) is highlighted in both the graphics pane and the browser.
A **double-click** on an element displays a dialog for editing its properties. (Pressing Esc, or the OK or Cancel button in the dialog, subsequently closes the dialog.)

A **right-click** anywhere in the interface displays a context-sensitive menu (also referred to as a *context menu*). The contents of a context menu depend on the position of the cursor when you click.

![Context Menu Example](image)

*If you right-click on an element, a menu is displayed containing actions you can take for that element.*

---

**Note:** GoldSim utilizes right-click context menus extensively. In fact, you may come to rely almost solely on this technique to carry out most GoldSim actions. If you want to do something with an object, and are in doubt about how to accomplish the action, right-click!

---

**Moving an object:** You can move an object in the graphics pane by *dragging* it from one location to another. Note, however, that you cannot move an element from or to the browser.

**Displaying a tool-tip:** You can display a tool-tip for a model object in the graphics pane and browser by holding the cursor over the object (without clicking).

**Selecting multiple objects:** You can select multiple objects in the graphics pane by Ctrl-clicking on multiple objects in succession or by creating a "selection box" by *dragging* your cursor from one point in the graphics pane (not on an object) to another point.
Saving, Opening, and Closing GoldSim Files

GoldSim models are saved in files with the extension gsm (e.g., mymodel.gsm). GoldSim does not produce separate input and output files. The model definition and the results generated by the model are all stored in one file – the gsm (GoldSim Model) file.

The Standard toolbar provides buttons for saving and opening files:

- **New**
- **Open**
- **Save**

To save a file, press **Ctrl+S**, press the **Save** button on the Standard toolbar, or select **File | Save** from the menu bar.

The first time that you save a new file, you will be prompted for the file name.

GoldSim can have only one model file open at a time (although you can have multiple instances of GoldSim running simultaneously, each with a different model file).

If you select **File | Save As** from the menu bar, GoldSim saves the current model with a new (user-specified) name, keeping the newly named file open. Selecting **File | Save Copy As** from the menu bar also saves the file to a new (user-specified) name, but keeps the original file open.

If you have made changes to a file but have not yet saved it, this will be indicated by an asterisk (*) next to the file name in the title bar for the GoldSim window.

To close a file (without exiting GoldSim), you can either open a new file or open an existing file. When you do so, if the current file has been changed, you will be prompted to save it before the new file is loaded.

To open a new GoldSim file, press the **New** button on the Standard toolbar, press **Ctrl+N**, or select **File | New** from the menu bar.
To open an existing file, press the **Open** button on the Standard toolbar, press **Ctrl+O**, or select **File | Open** from the menu bar. You will be prompted for a file name. To open an existing file, press the **Open File** button on the Standard toolbar, press **Ctrl+O**, or select **File | Open** from the menu bar. You will be prompted for a file name.

You can also open previous files you have saved by selecting one from the **File** menu.

The most recently saved files are listed at the bottom of the **File** menu.

---

**Restoring Files After an Unexpected Failure Using Auto-Save**

GoldSim allows you to recover a copy of your model file should GoldSim unexpectedly terminate for some reason while your file is open. This would allow you to restore changes you made to the file since you last manually saved it.

---

**Note:** Obviously, GoldSim should not terminate unexpectedly. If it ever does, please contact the GoldSim Technology Group at through the GoldSim Help Center (https://goldsim.zendesk.com).

The feature works by saving a copy of your model at a user-specified frequency (by default, every 10 minutes) to a separate location. In particular, the recovery file is saved temporarily in the local AppData directory of your computer (i.e., it does not overwrite the original file). If GoldSim encounters a problem and closes abruptly (i.e., the software crashes), Windows will automatically restart GoldSim, and you will be given an opportunity to view the auto-saved recovery file. If you had made a number of changes and had not saved your original file for some time prior to the crash, the recovery file may represent a “newer” version of the file than the original file. When GoldSim restarts, you will be given the opportunity to replace your original file with the recovery file. If you choose not to use the recovery file, it will be automatically deleted (until it is regenerated with the next auto-save).

To understand how the auto-save feature works, it is useful to walk through the dialogs generated by GoldSim during this process.
If GoldSim unexpectedly terminates, Windows will automatically restart the software, and you will be presented with the following screen:

It is critical that you do not press **Cancel** when you see this (or the next) dialog. If you do, your auto-saved recovery file will be deleted!

A second dialog will then appear, indicating that GoldSim is looking for a recovery file.

Finally, once the file is located, the following dialog will be displayed:

Note that the dialog will tell you if the recovered model file is more recent than the original file. If you choose “No”, the recovered file will be deleted and a new file will be opened.

If you select “Yes”, the recovered file will be opened. In the Title Bar, you will notice that “[recovered]” will be appended to the end of the filename:

If you choose to Save, you will be prompted for a filename. You can save this model with the same name as the original (to overwrite it) or save it with a new name. Once that is done, the auto-saved model will be deleted from Apps Data directory on your computer.

The frequency at which the model is auto-saved can be controlled from the **General** tab of the Options dialog (accessed via **Model | Options...** from the main menu):
By default, the **Save crash recovery information every X minutes** is checked on and set to 10 minutes.

Several points should be noted regarding the auto-save feature:

- Auto-save files are deleted whenever a file is saved or closed.
- This feature is only available on systems using Windows Vista and higher.
- This feature is not available for the GoldSim Player.
- The file will only be auto-saved if the file is in Edit mode and the application is in an idle state (e.g., a dialog is not open). Otherwise the save will be queued.

**Note:** The auto-saved file is saved in the GoldSim sub-directory of the AppData directory for the local user of the system in which GoldSim is Running (e.g., C:UsersusernameAppDataLocalGTGGoldSim). You can quickly find this location by opening Windows File Explorer and typing the keyword %LocalAppData% into the address bar.

**Password-Protecting a Model File**

In some instances, you may wish to protect a model file with a password, such that in order for someone to open it, they must provide a password.

You can do this by selecting **File → Protect…** from the main menu. When you do this, the following dialog will appear:
The password must have a length of 4 to 20 characters.

After you do this, anyone who tries to subsequently open the model will be presented with the following dialog:

Once a file is opened, you can remove the password by selecting File|Protect… from the main menu. If the file is protected, it will prompt you for the password again. Once you enter it, the Protect Model File dialog will be displayed (this time with the password):

To remove the password-protection, simply clear both fields and press OK.

Note that once a file is password-protected, users will be prompted for a password under the following circumstances:

- When the user tries to open the file;
- When the user tries to change the password (by accessing File|Protect…);
- When the model is used to create a Player file, and the Player file is opened;
- When the model is imported as a SubModel; and
- When the model is opened using command line arguments.

**Warning:** If you forget the password, there is no way to recover the file (as providing any kind of “backdoor” would weaken the security). So if you choose to use this feature, make sure you record the password!

If you wish to send a model to someone else via email, you can quickly do so by selecting File|Send to…. GoldSim will open your default email program, and attach the model file to a new message.

At any given time, a GoldSim model is in one of four states (referred to as modes):

**Edit Mode:** The model is currently being edited and does not contain any simulation results.
**Run Mode:** The model is currently running. In this mode, you can pause the simulation and view the model, but model objects cannot be edited (although graphical objects can be changed).

**Result Mode:** The model has been run and contains simulation results. In this mode, you can navigate the model and view results, but model objects cannot be edited (although graphical objects can be changed).

**Scenario Mode:** This is a special mode (similar in some respects to Result Mode) that only exists if you are using GoldSim’s scenario feature. To be in this mode, the model must contain at least one scenario that has been defined by the user, and at least one scenario must have scenario results. In this mode, you can navigate the model and compare scenario results, but no elements (other than Result elements and Scenario Data elements) can be added, deleted or edited.

*Read more: Understanding Simulation Modes* (page 498); *Carrying Out a Simulation (Run Mode)* (page 499); *Viewing Results (Result Mode)* (page 504); *Creating, Running and Comparing Scenarios* (page 504).

The mode that the model is in is clearly identified in the status bar (in the lower left-hand corner of the GoldSim window). In addition, the status bar takes on a different color in each mode:

- **Edit Mode:** The status bar is blue.
- **Run Mode:** The status bar is pink.
- **Run Mode, but is paused:** The status bar is rust brown.
- **Result Mode:** The status bar is green.
- **Scenario Mode:** The status bar is orange.

You can only edit a model when it is in Edit Mode (and to a very limited extent, in Scenario Mode). When you create a new model file, it is in Edit Mode until you run it for the first time.

If you are viewing a model that is in Result or Run mode, editing will be locked (e.g., most of the property dialogs and menu items will be grayed out), although you can edit graphical objects such as text and images.

## Creating Elements and Links

Elements are the basic building blocks of a GoldSim model. GoldSim models are built by connecting the outputs of one (or more) elements to the inputs of other elements. These connections are referred to as links. A complex model can have thousands of elements and links.
GoldSim Element Types

GoldSim provides a wide variety of built-in elements for entering data and manipulating variables. Each element represents a building block of the model, and has a default symbol by which it is represented in the graphics pane and an icon by which it is represented in browsers and menus.

The GoldSim elements can be divided into the following categories:

- Input elements
- Stock elements
- Function elements
- Event elements
- Delay elements
- Result elements

An additional type of element, the Container, is used to hierarchically organize other elements.

The basic element types, their functions, and their default graphic symbols and icons are summarized in the following sections.

Note: The symbols used for elements in the graphics pane can be customized by the user. However, the icons (which appear in the browsers) cannot be changed.

Overview of Input Elements

Input elements are used to create inputs (exogenous data) that subsequently define the properties of other elements.

<table>
<thead>
<tr>
<th>Element</th>
<th>Default Symbol</th>
<th>Browser Icon</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td></td>
<td><img src="image" alt="Data Icon" /></td>
<td>Defines scalar, vector or matrix data.</td>
</tr>
<tr>
<td>Stochastic</td>
<td><img src="image" alt="Stochastic Icon" /></td>
<td></td>
<td>Defines uncertain data as probability distributions.</td>
</tr>
<tr>
<td>Time Series</td>
<td><img src="image" alt="Time Series Icon" /></td>
<td></td>
<td>Accepts time histories of data and converts them to an appropriate form for use in the model.</td>
</tr>
<tr>
<td>Lookup Table</td>
<td><img src="image" alt="Lookup Table Icon" /></td>
<td></td>
<td>Defines a one, two or three dimensional lookup table (response surface).</td>
</tr>
<tr>
<td>History Generator</td>
<td><img src="image" alt="History Generator Icon" /></td>
<td></td>
<td>Generates stochastic time histories based on specified statistics.</td>
</tr>
</tbody>
</table>

---
Overview of Stock Elements

Stocks elements are elements which impart inertia and memory to a system. These kinds of elements are responsible for internally generating the dynamic behavior of a system. At any point in time in a simulation, the outputs of stock elements are computed based on the historical values of their inputs. Mathematically, stocks represent time integrals.

<table>
<thead>
<tr>
<th>Element</th>
<th>Default Symbol</th>
<th>Browser Icon</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integrator</td>
<td></td>
<td><img src="image" alt="Integrator Icon" /></td>
<td>Integrates values. Also can compute moving average of a specified input.</td>
</tr>
<tr>
<td>Reservoir</td>
<td></td>
<td><img src="image" alt="Reservoir Icon" /></td>
<td>Integrates (and conserves) flows of materials, allowing for upper and lower bounds to be specified.</td>
</tr>
<tr>
<td>Pool</td>
<td></td>
<td><img src="image" alt="Pool Icon" /></td>
<td>More complex version of Reservoir, making it easier to model multiple inflows and outflows.</td>
</tr>
</tbody>
</table>

Read more: Using Stock Elements (page 228).

Overview of Function Elements

Function elements are elements that manipulate and transform information or material. At any point in time in a simulation, the outputs of these elements are computed based on the current values of their inputs.

<table>
<thead>
<tr>
<th>Element</th>
<th>Default Symbol</th>
<th>Browser Icon</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expression</td>
<td><img src="image" alt="Expression Icon" /></td>
<td></td>
<td>Defines mathematical or logical expressions.</td>
</tr>
<tr>
<td>Script</td>
<td><img src="image" alt="Script Icon" /></td>
<td></td>
<td>Allows you to create your own custom element using a simple procedural language.</td>
</tr>
<tr>
<td>Previous Value</td>
<td><img src="image" alt="Previous Value Icon" /></td>
<td></td>
<td>Returns the value of its input from the previous model update.</td>
</tr>
<tr>
<td>Extrema</td>
<td><img src="image" alt="Extrema Icon" /></td>
<td></td>
<td>Computes the highest (peak) or lowest (valley) value achieved by its input.</td>
</tr>
</tbody>
</table>
# Creating Elements and Links

<table>
<thead>
<tr>
<th>Element</th>
<th>Default Symbol</th>
<th>Browser Icon</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selector</td>
<td><img src="image" alt="Selector Icon" /></td>
<td><img src="image" alt="Selector Icon" /></td>
<td>Defines expressions with nested if…then logic.</td>
</tr>
<tr>
<td>Splitter</td>
<td><img src="image" alt="Splitter Icon" /></td>
<td><img src="image" alt="Splitter Icon" /></td>
<td>Splits an incoming signal into multiple outputs based on specified fractions.</td>
</tr>
<tr>
<td>Allocator</td>
<td><img src="image" alt="Allocator Icon" /></td>
<td><img src="image" alt="Allocator Icon" /></td>
<td>Allocates an incoming signal into multiple outputs given specified demands and priorities.</td>
</tr>
<tr>
<td>Sum</td>
<td><img src="image" alt="Sum Icon" /></td>
<td><img src="image" alt="Sum Icon" /></td>
<td>Facilities the addition of multiple values.</td>
</tr>
<tr>
<td>Convolution</td>
<td><img src="image" alt="Convolution Icon" /></td>
<td><img src="image" alt="Convolution Icon" /></td>
<td>Solves a convolution integral.</td>
</tr>
<tr>
<td>And</td>
<td><img src="image" alt="And Icon" /></td>
<td><img src="image" alt="And Icon" /></td>
<td>Combines multiple conditions using logical AND.</td>
</tr>
<tr>
<td>Or</td>
<td><img src="image" alt="Or Icon" /></td>
<td><img src="image" alt="Or Icon" /></td>
<td>Combines multiple conditions using logical OR.</td>
</tr>
<tr>
<td>Not</td>
<td><img src="image" alt="Not Icon" /></td>
<td><img src="image" alt="Not Icon" /></td>
<td>Logical NOT.</td>
</tr>
</tbody>
</table>

**Read more:** [Using Basic Function Elements](#) (page 274); [Script Elements](#) (page 860); [Solving Convolution Integrals](#) (page 817).

A special subcategory of function elements can be used to link GoldSim to external applications:

<table>
<thead>
<tr>
<th>Element</th>
<th>Default Symbol</th>
<th>Browser Icon</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>External (DLL)</td>
<td><img src="image" alt="External Icon" /></td>
<td><img src="image" alt="External Icon" /></td>
<td>Dynamically links to a user-specified external function (compiled as a DLL).</td>
</tr>
</tbody>
</table>
Creating Elements and Links

<table>
<thead>
<tr>
<th>Element</th>
<th>Default Symbol</th>
<th>Browser Icon</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spreadsheet</td>
<td>![Spreadsheet Icon]</td>
<td>![Spreadsheet Icon]</td>
<td>Dynamically links to an Excel spreadsheet.</td>
</tr>
<tr>
<td>File</td>
<td>![File Icon]</td>
<td>![File Icon]</td>
<td>Dynamically copies a file to a specified directory (for use by External elements).</td>
</tr>
</tbody>
</table>

**Read more:** [Using External Application Elements](#) (page 910).

Another category of elements allows you to superimpose the occurrence and effects of discrete events onto continuously varying systems:

<table>
<thead>
<tr>
<th>Element</th>
<th>Default Symbol</th>
<th>Browser Icon</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timed Event</td>
<td>![Timed Event Icon]</td>
<td>![Timed Event Icon]</td>
<td>Generates discrete event signals based on a specified rate of occurrence, regularly or according to a specified distribution (i.e., randomly).</td>
</tr>
<tr>
<td>Triggered Event</td>
<td>![Triggered Event Icon]</td>
<td>![Triggered Event Icon]</td>
<td>Generates discrete event signals based on one or more specified conditions.</td>
</tr>
<tr>
<td>Decision</td>
<td>![Decision Icon]</td>
<td>![Decision Icon]</td>
<td>Generates one of up to three defined discrete event signals based on specified conditions.</td>
</tr>
<tr>
<td>Random Choice</td>
<td>![Random Choice Icon]</td>
<td>![Random Choice Icon]</td>
<td>Generates a user defined discrete event signal based on specified probabilities.</td>
</tr>
<tr>
<td>Milestone</td>
<td>![Milestone Icon]</td>
<td>![Milestone Icon]</td>
<td>Records the time at which a particular event or specified condition(s) occurs.</td>
</tr>
<tr>
<td>Status</td>
<td>![Status Icon]</td>
<td>![Status Icon]</td>
<td>Generates a condition (True/False) in response to particular events or specified conditions.</td>
</tr>
<tr>
<td>Discrete Change</td>
<td>![Discrete Change Icon]</td>
<td>![Discrete Change Icon]</td>
<td>Generates a discrete change signal (a value) that can subsequently discretely modify the values of other elements (e.g., Integrators and Reservoirs).</td>
</tr>
<tr>
<td>Interrupt</td>
<td>![Interrupt Icon]</td>
<td>![Interrupt Icon]</td>
<td>Interrupts a simulation when a specified event or condition occurs and displays a user-defined message and/or writes a message to the run log.</td>
</tr>
</tbody>
</table>
Overview of Delay Elements

Read more: Chapter 5: Simulating Discrete Events (page 351).

Delay elements represent a type of element which imparts inertia and memory to a system by delaying (and optionally dispersing) continuous and discrete signals or flows. Like Stocks, at any point in time in a simulation, the outputs of Delays are computed based on the historical values of their inputs.

<table>
<thead>
<tr>
<th>Element</th>
<th>Default Symbol</th>
<th>Browser Icon</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Material Delay</td>
<td></td>
<td></td>
<td>Delays (and optionally disperses) material flows.</td>
</tr>
<tr>
<td>Information Delay</td>
<td></td>
<td></td>
<td>Delays (and optionally disperses) a signal.</td>
</tr>
<tr>
<td>Event Delay</td>
<td></td>
<td></td>
<td>Delays a discrete event signal. Can also be used to simulate queues.</td>
</tr>
<tr>
<td>Discrete Change Delay</td>
<td></td>
<td></td>
<td>Delays a discrete change signal. Can also be used to simulate queues.</td>
</tr>
</tbody>
</table>

Overview of Result Elements

Read more: Using Delay Elements (page 320).

Result elements provide a convenient and powerful mechanism to collect, analyze and display simulation results.

<table>
<thead>
<tr>
<th>Element</th>
<th>Default Symbol</th>
<th>Browser Icon</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time History Result</td>
<td></td>
<td></td>
<td>Displays time history results.</td>
</tr>
<tr>
<td>Distribution Result</td>
<td></td>
<td></td>
<td>Displays probability distributions.</td>
</tr>
<tr>
<td>Multi-Variate Result</td>
<td></td>
<td></td>
<td>Displays multi-variate results (scatter plots, correlation tables, raw data).</td>
</tr>
<tr>
<td>Array Result</td>
<td></td>
<td></td>
<td>Displays vectors and matrices of results.</td>
</tr>
</tbody>
</table>
Creating Elements

There are four ways to create (insert new) elements in GoldSim:

- using the context menu of the graphics pane;
- using the element toolbar;
- using the context menu for a Container element; and
- using the main menu.

The first two methods are the most common and straightforward method, and they also have the advantage that they allow you to specify exactly where the element is inserted in the graphics pane.

Inserting an Element
Using the Context Menu
of the Graphics Pane

The steps involved to insert a new element using the context menu of the graphics pane are as follows:

1. Right-click anywhere in the graphics pane away from an object (i.e., in an empty part of the screen). A context-sensitive menu is displayed.
2. Place your cursor over the Insert Basic Element item to expand the menu. All of the GoldSim elements are listed by category.
3. Click on the element type that you wish to insert.
4. The editing dialog for the element is displayed.
5. Enter the properties for the element and press OK.
6. The element will be inserted in the graphics pane with its upper-left corner at the location of the cursor when you right-clicked.

Inserting an Element
Using the Element Toolbar

One of the toolbars provided by GoldSim at the top of the screen is the Element toolbar. Each button in the toolbar represents a different element category:

The steps involved to insert a new element using the context menu of the graphics pane are as follows:

1. Left-click on the button corresponding to the element category of interest. A menu will be displayed showing all of the elements in that category:
Creating Elements and Links

Chapter 3: Building a Model in GoldSim

GoldSim User's Guide

2. Left-click on the element type that you wish to insert. When you do so, the cursor will change and will look like this:

3. Move the cursor to where you wish to place the element and left-click.

4. The editing dialog for the element is displayed.

5. Enter the properties for the element and press OK.

6. The element will be inserted in the graphics pane with its upper-left corner at the location of the cursor when you left-clicked.

Although the most common way to insert a new element is to use the context menu in the graphics pane, you can also insert an element by using the context menu of a Container element or by selecting Model | Insert Element from the main menu. In both cases, you will be presented with a menu from which you can select an element to insert.

When using these two methods, you cannot control where on the graphics pane the element is inserted. It will always be inserted in the upper left-hand corner of the graphics pane.

If the context menu of a Container is used, the element will be inserted in the selected Container. If the main menu is used, the element will be inserted in the current Container.

Read more: Understanding Containers (page 95).

GoldSim models are built by linking the outputs of one (or more) elements to the inputs of other elements. Most GoldSim elements have at least one input and one output.

The easiest way to see the inputs and outputs is to use the input and output ports. By default, ports are hidden until you hold your cursor over the element.

Note: Ports are also automatically shown under some other circumstances even without holding your cursor over the element (e.g., if the element has results). You can also optionally control when ports are displayed. Ports are discussed in more detail in the next section.

Read more: Understanding Input and Output Ports (page 76).

In all cases, however, holding your cursor over an element will cause the ports to be displayed. The ports appear as small squares at the bottom corners of the element:
Placing your cursor over a port changes the appearance of the cursor to a pointing hand. If you left-click on the port with this cursor, a browser view window (referred to as an input or output interface) is displayed.

The input interface displays the element's inputs:

The output interface displays the element's outputs:

**Note:** You can also view the inputs and outputs in the browser.

**Read more:** Using the Browser (page 105).

As can be seen above, inputs and outputs have distinct names. Input names are “hard-wired” for each element type and cannot be changed by the user (e.g., the input names for all Reservoir elements are the same: Initial Value, Addition Rate, Withdrawal Rate Request, etc.). These input names correspond to the labels for the input fields in the property dialog for the element.

Some elements, such as an Expression, have a single output. Other elements, such as a Reservoir, have multiple outputs. If an element has multiple outputs, one of the outputs is usually defined as the *primary output* for the element. Any other output for the element is referred to as a *secondary output*.

The name of the primary output of an element is always the same as the name of the element itself. Hence, the name for the primary output of the Reservoir element shown above is “Pond”, the same as the name of the element. The output names for secondary outputs are “hard-wired” for each element type and
cannot be changed by the user. In the example above, “Overflow_Rate”, “Is_Full” and “Withdrawal_Rate” are secondary outputs.

Output names are important, because links between elements are created by referencing the name of the output of one element in an input field of another element.

Read more: Referencing Outputs of Other Elements (Creating Links) (page 80).

Most GoldSim elements have both input and output ports. (A small number of elements have only one or neither.)

Holding your cursor over an element will cause the ports to be displayed. The ports appear as small squares at the bottom corners of the element:

Ports serve two important purposes:

1. By left-clicking on them, you can view a list of the inputs and outputs for the element:

There are several reasons to do this, with the most common one being viewing the list of outputs in order to display results for a particular output.

Read more: Viewing Results (page 118).

2. By changing their appearance, ports are used to convey status information regarding the element and provide important visual cues when navigating a model. In particular,

- When in Result Mode, if an element has results, the output port becomes green:

Read more: Viewing Results (page 118).

- If an input or output of the element is linked to an element in another Container, a dot appears in the middle of the input and/or output port:
Pond1 has an input that comes from another Container. Pond2 has an input that comes from another Container and has an output that is used in another Container.

**Read more:** [Influences Between Containers](page 99).

**Note:** There is a third instance in which the ports change their appearance. In particular, if an element has a manually “forced” precedence to address an ambiguous causality sequence, the input port becomes gold. This, however, is a highly advanced feature and would rarely be encountered.

**Read more:** [Addressing Ambiguous Causality Sequences](page 971).

The ports for an element are always displayed when you hold your cursor over it. However, you can control how ports are displayed when the cursor is not being held over an element. This is controlled via the main menu by selecting **View | Show Element Ports**:

As can be seen, there are three options:

- **Smart.** This is the default. If this is selected, ports are only shown if they have some status information to display (e.g., results, links to other Containers). In the examples shown above, this is why only one of the two ports are shown in some cases.
- **All.** The ports are always shown.
- **None.** None of the ports are shown.

In the overwhelming majority of cases, the default (**Smart**) is the appropriate option. Occasionally, you may want to hide all ports (**None**) so as to make the graphics pane look less complicated. It is rare that you would want to show **All**.

**Note:** Changes to the appearance of the ports are not saved with the file. Every time you reopen the file, the appearance returns to the default (**Smart**).

**Editing an Element’s Properties and Creating Links**

The properties of an element can be edited either when it is first created, or by double-clicking on it at any time (or by selecting the element and pressing **Alt-Enter**).
The properties dialog for a typical element (in this case, an Integrator) is shown below:

With a few special exceptions (e.g., Containers, Result elements), all elements have the following fields or buttons:

- **Element ID.** The name of the element
- **Description.** A brief (255 characters) description of the element.
- **Appearance button.** Provides access to a dialog for changing the graphical appearance of the element.
- **Display Units.** The units (which determine the dimensions) of the element's primary output.
- **Type button.** Provides access to a dialog for editing the type (value or condition) and order (scalar, vector, matrix) of the element's primary output.
- **Save Results checkboxes.** Used to specify whether or not results are saved for the element.
- **Definition fields.** One or more input fields that define the properties of the element.

The default ID (name) for a new element is the element type followed by a number (e.g., Expression1, Integrator3). You can, of course, change the name.

**Note:** Element names can only include letters, numbers, and the underscore (“_”) character. They cannot include spaces and must begin with a letter. In addition, some names (e.g., sin, cos, min) are reserved for use as functions and cannot be used as an element name.

**Note:** Two types of elements (Result elements and Dashboards) have slightly more flexibility in terms of their names. In particular, their names can include spaces.
In addition to changing the element's name in the dialog, you can also edit it by Ctrl+double-clicking on the name of the element in the graphics pane. If you Ctrl+double-click on the element name in the graphics pane, it will become selected and available for editing (indicated by the dark background).

**Note:** By default, element names are forced to be unique (i.e., the elements are global). For most applications, however, you will want to build models which have elements with non-unique names. In order to do so, the each identically-named elements must have a different **scope**. You can use a feature called localized Containers to create local scopes.

**Read more:** [Localizing Containers](#) (page 944).

You should enter a **Description** for the element below the **ID** in the properties page. Descriptions can be up to 255 characters long. The Description field shows two lines. If you enter more than two lines of text, scroll bars appear. While entering a Description, you can use **Ctrl+Return** to insert a line break.

Note that the Description appears in the tool-tip displayed for the element in the graphics pane and the browser.

A tool-tip is displayed when you hold the cursor over the element in the graphics pane. The description is always displayed in the tool-tip below the element name.

Adding descriptions for every element makes your model much more transparent and easy to understand. It is strongly recommended that you add a description at the time you create the element. In larger models, it is highly unlikely that you will go back and add descriptions to the elements after you have completed the model.

**Note:** Descriptions are intended to be relatively short (255 characters). You can add very detailed documentation to an element by attaching a Note to it.

**Read more:** [Creating, Editing and Viewing Notes](#) (page 755).

You can access a menu for changing the graphical appearance of an element by pressing the **Appearance**… button.

**Read more:** [Editing the Appearance of Elements](#) (page 434).

**Specifying the Contents of an Input Field**

You specify the contents of an input field in a properties dialog in one of three ways:

- You type in a number;
- You create a link to another element by entering the name of an output of that element; or
Creating Elements and Links

- You enter a mathematical expression, such as 3*\log(250). Note that the expression itself can incorporate links (the names of element outputs), such as A*\log(B), where A and B are output names.

Examples of all three of these types of inputs are shown below:

| Definition | 4.5 | number |
| Initial Value: | 3 \times \frac{A}{B} | expression (with links) |
| Rate of Change: | accident | link to another element |
| Discrete Change: |

Referencing Outputs of Other Elements (Creating Links)

Some elements, such as an Integrator, have a single output, while other elements, such as a Stochastic or a Reservoir, have multiple outputs. If an element has multiple outputs, one of the outputs is usually defined as the primary output for the element. Any other outputs for the element are referred to as secondary outputs.

Read more: Element Inputs and Outputs (page 74).

The name of the primary output of an element is always the same as the name of the element itself. Output names are important because links between elements are created by referencing the name of the output of one element in one of the input fields for another element. In the example above, A, B, and Accident are the names of the primary outputs of the elements A, B and Accident, respectively.

The names for secondary outputs are "hard-wired" for each element type. For example, a Stochastic always has two secondary outputs named “Probability_Density” and “Cumulative_Probability”. As a result, when referencing the secondary output of an element, it is not sufficient to simply enter the output name, as GoldSim would not be able to distinguish between the outputs if there were multiple occurrences of that type of element in the model (e.g., if there were two Stochastics in the model, GoldSim would not know which element was being referred to).

To avoid this problem, an output which is not the primary output is referenced by using both the element name and the output name: elementID.outputID. For example, the two secondary outputs of the Stochastic element named B would be referenced as:

B.Probability_Density

and

B.Cumulative_Probability.

Links and Influences

When a link is made between the output of one element and the input of another element in the same Container, GoldSim automatically draws an arrow connecting the two elements in the graphics pane:
Creating Elements and Links

These arrows are referred to as influences.

Read more: Understanding Influences (page 98).

When you place your cursor into an input field and begin to type, you will notice that GoldSim color-codes the text as you type. Red underlined text indicates that GoldSim cannot interpret that portion of the expression (e.g., due to incorrect syntax). Once the item becomes valid, it will no longer be shown in red and underlined. If you leave an input field that is invalid (e.g., change the focus to another field), the entire field will be highlighted in pink.

Read more: Error Checking in Input Fields (page 93).

Note: In addition to displaying errors and color-coding text as you type, as you enter characters into an input field, GoldSim automatically suggests the names of existing outputs (that are consistent with those characters) that you may want to link to at the bottom of the input field.

Read more: Displaying Link Suggestions in Input Fields (page 83).

When you first place the focus in (i.e., click into) an input field (or if you stop typing within an input field), a tool-tip is displayed. If the expression is invalid, the tool-tip will explain why; if the expression is valid, it will display its current value. If you start typing (or move the focus to another field), the tool-tip will disappear. If you stop typing, the tool-tip will reappear. You can also view the tool-tip for an input field by simply holding the cursor over the field (while the focus is elsewhere). The tool-tip disappears when you move the cursor away.

The table below summarizes the mathematical and relational operators that are supported by GoldSim for use within expressions in input fields, along with their precedence:

<table>
<thead>
<tr>
<th>Precedence</th>
<th>Type</th>
<th>Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highest</td>
<td>parentheses</td>
<td>()</td>
</tr>
<tr>
<td></td>
<td>exponetiation</td>
<td>**, ^</td>
</tr>
<tr>
<td></td>
<td>unary minus</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>multiplication, division</td>
<td>*, /</td>
</tr>
<tr>
<td></td>
<td>addition, subtraction</td>
<td>+, -</td>
</tr>
<tr>
<td></td>
<td>relational test</td>
<td>&gt;, &lt;, &gt;=, &lt;=</td>
</tr>
<tr>
<td>Lowest</td>
<td>equality test</td>
<td>=, ==, &lt;&gt;, !=</td>
</tr>
</tbody>
</table>

The precedence of the operators determines the manner in which expressions are interpreted and evaluated. For example, in the expression:
Creating Elements and Links

4 + 3 * 6

3 times 6 would be evaluated first, then 4 would be added to that quantity (resulting in a value of 22) because the “*” operator has a higher precedence than “+” operator. However, (4 + 3) * 6 would result in a different value (42) because the parentheses have higher precedence than the “*” operator.

**Warning:** The precedence of the unary minus as listed in the table above (lower precedence than exponentiation) only applies when it precedes a variable (as opposed to a number). That is, the expression \(-X^2\), where \(X=3\) would be interpreted as \(-9\). However, if a unary minus directly precedes a number, it is always considered to be part of the number itself, and hence the operator has a higher precedence than exponentiation. That is, the expression \(-3^2\) would be interpreted as 9.

**Note:** The equality (== and =) and inequality (< > and !=) operators assume that two values are identical if they are the same to 12 significant figures. Because double-precision arithmetic (used in the GoldSim code) has a precision of about 15 significant figures, random roundoff errors should generally not become significant. However, there is a caveat to this rule: if GoldSim compares a non-zero value to a zero value, they are never treated as identical, even if the non-zero value is extremely small. For example, if \(A\) was exactly 1 and \(B\) was different from 1 at the 14th significant figure, then \(A=B\) would be true, but \(A-B=0\) would be false.

In addition to the operators listed above, GoldSim also provides a number of built-in functions (e.g., sin, cos, log) and constants (e.g., \(\pi\), \(e\), the gravitational constant, Avogadro’s number) which can be used to build expressions. GoldSim also provides three logical operators (And, Or, Not) which only operate on conditions.

**Read more:** [Understanding Output Attributes](#); [Creating Conditional Expressions](#); [Entering Mathematical Expressions into Element Input Fields](#).

When creating expressions, spaces are ignored. For example, the expressions “4*5” and “4 * 5” would both be interpreted as 20. You should feel free to add spaces to your expressions in order to improve their readability. The two exceptions to this rule are 1) when you are adding unit strings to expressions (e.g., m/sec, $/day) spaces within the unit string are not allowed; and 2) when you are using a unary minus with a number (e.g., -3C), spaces between the minus sign and the number are not allowed.

**Read more:** [Using Units in Element Input Fields](#).

GoldSim color-codes expressions to make them easier to read. The colors used are:

- **Black:** Numbers and functions
- **Green:** Output names (links) and functions
- **Gray:** Operators
- **Blue:** Units
In addition, if you use nested sets of parentheses in an expression, when you click inside the expression, GoldSim will highlight the nearest matching pair.

If the expression that you type into an input field is longer than the field in the dialog, the field will automatically expand when it has the focus such that the entire expression can be viewed.

\[
(3.4 \times \text{Output1}) + (5.4 \times \text{Output2}) + (6.2 \times \text{Output3}) + (3.3 \times \text{Output4}) + (6.5 \times \text{Output3})
\]

**Note:** The input field for an Expression element is larger than for other elements (i.e., it is always expanded and hence accommodates several rows without requiring the focus).

### Divide By Zero Errors in Input Fields

In some cases, you may enter an expression into an input field such as X/Y where at some point during the simulation, both X and Y are zero. In most instances, when GoldSim encounters zero in the denominator of an expression, it will display a fatal error and stop the simulation. However, if both the numerator and the denominator are zero, GoldSim will not display a fatal error; instead, the resulting expression will be set to zero and the simulation will continue. This can be very convenient, as it allows you to avoid adding complex logic to check for such cases.

Note, however, that GoldSim only sets the expression to zero if both the numerator and denominator are zero. To understand this, consider an example where A and C are equal to zero, and B is non-zero. In this case, the following two expressions would be treated differently by GoldSim:

- A*B/C: would return zero, as A*B would be evaluated first (and considered to be the numerator for C).
- A*(B/C): would result in a fatal error, as B/C would be evaluated first, and B is non-zero.

### Displaying Link Suggestions in Input Fields

When editing an input field, by default GoldSim automatically suggests the names of existing outputs at the bottom of the input field. That is, when you place your cursor into an input field and begin to type, GoldSim provides a list of suggestions for output names that are consistent with what you have entered so far.

For example, assume that the outputs A1, A2 and A3 all exist in your model (at a scope that is accessible from the current input field), and within an input field you type the letter A. The following will be displayed at the bottom of the input field:
If you simply want to select the first item in the list, you can press the **Tab** key. The suggestion box will close, the selection will be inserted, and in the input field, the cursor is placed to the immediate right of the selected output.

You can select one of the other suggestions either by selecting it with your mouse or using the **Up** and **Down** arrow keys to select an option.

If you use the mouse to make the selection, the suggestion box will close, the selection will be inserted, and in the input field, the cursor is placed to the immediate right of the selected output.

If you use the **Up** and **Down** arrows to make the selection, then the following rules apply:

- You can continue to move through the list using the **Home**, **End**, **PgUp** and **PgDn** keys.
- If you press the **Tab** key or right arrow, the suggestion box will close, the selection will be inserted, and in the input field, the cursor is placed to the immediate right of the selected output.
- If you press a non-character key (such as an operator like +), the suggestion box will close, the selection will be inserted, and in the input field, the operator is inserted and the cursor is placed to the immediate right of the inserted operator.

The suggestion list will also be closed if you left or right click elsewhere in the input field, or press **Esc**.

The following should be noted regarding how GoldSim constructs the suggestion list:

- The list only contains items that are within the scope of the input field.
  
  **Read more:** Localizing Containers (page 944).

- All locally available variables are included in the list (you can see these by typing ~):

  ![Local Variables](image)

  **Read more:** Understanding Locally Available Properties (page 808).

- If you wish to link to a secondary output, you can do so by typing a period after the element name in the input field. The suggestion box will then list any secondary outputs that exist for that element:

  ![Secondary Outputs](image)

  **Read more:** Referencing Outputs of Other Elements (Creating Links) (page 80).
• Similarly, if you wish to link to an exposed output of a localized Container, you can do so by typing a period after the name of the Container in the input field. The suggestion box will then list any exposed outputs that exist for that Container.

Read more: Localizing Containers (page 944).

• Pressing Ctrl+Space causes the suggestion box to be displayed. If the cursor is to the immediate right of a character, then the list that is consistent with the preceding character string is shown. If the cursor is not to the immediate right of a character (e.g., if there is a space or operator before the cursor), then a list of ALL available outputs is provided.

Input Field Context Menu

Right-clicking within an input field (or pressing Ctrl+M) displays a context menu to assist you in adding links and creating expressions:

Selecting Insert Link… from this menu (or pressing Ctrl+I directly from the input field) displays a browser tree showing all of the elements in the model:
Note: For some elements, an input field may only accept a limited class of links, and will not allow the user to enter an expression. In these cases, the Insert Link dialog will only display a limited subset of the elements in the model (those that can actually be linked to this particular input).

This tree is organized by containment (and in alphabetical order) in the same manner as the browser. To insert a link from this dialog, you select a specific output object (or an element having a primary output), and then press OK (or simply double-click on the output). The output name will be inserted into the expression.

Note: The Previous and Home buttons at the top of the dialog are very useful for rapidly finding elements to link to. Whenever you use the Insert Link dialog, GoldSim opens the tree in the directory containing the element to which you are linking (i.e., it assumes that in most instances you will want to link to a nearby element). GoldSim also remembers the Container of the last place that you inserted a link from. Pressing the Previous button opens the tree at this location. Pressing Home returns to the directory of the element to which you are linking. Holding the Ctrl key down while opening the dialog (i.e., selecting Insert Link) is equivalent to opening the dialog and immediately pressing the Previous button.

At the top of this browser is a search field to assist you in finding elements. Enter a string and press the Search button to the left of the string (the magnifying glass) or press F3 to find the next item in the browser that matches the search string. The search is not case-sensitive.

You can control where GoldSim searches by pressing Search Options..., which will display a dialog for selecting whether you want to search in IDs, Descriptions, and/or Notes.
Note: These Search options are stored in the Windows Registry (so that once you change them, they will remain until you edit them again, even when editing other GoldSim files).

The context menu for an input field can also be used to insert the built-in functions and constants provided by GoldSim, and to insert units into an expression.

Read more: Understanding Containers (page 95); Finding Elements (page 112); Entering Mathematical Expressions into Element Input Fields (page 124); Using Dimensions and Units (page 89).

Deleting Links and Influences

You cannot delete a link between two elements by simply deleting the influence. In fact, GoldSim provides no direct way to delete the influence (e.g., you cannot select it and press Delete).

The only way to delete a link (and hence the influence representing it) is to remove the output being referenced from the input field of the other element. For example, if Data element A is entered into one of the input fields for element B, a link will be created (and an influence will be drawn) between element A and element B. To delete this link (and influence), you must delete the reference to A in the input field for B.

Note, however, that it is possible to hide an influence (so that it is not visible in the graphics pane).

Read more: Filtering Influences (page 430).

Understanding Output Attributes

All element outputs in GoldSim have three output attributes: a type, an order, and dimensions:

**Type:** The output type is either a value or a condition. Values can take on any real number. Conditions have only two states: True or False.

**Order:** The order of an output can be scalar, vector or matrix. A scalar is a single item. A vector is a column (or row) of items. A matrix is a table (with rows and columns) of items.

**Dimensions:** Outputs can be assigned dimensions (e.g., length, volume, velocity) which can be displayed in any units that you choose (meters, gallons, feet/second). Dimensions can only be assigned to values (not to conditions).

Most GoldSim outputs default to being dimensionless, scalar values.

Output attributes are important because GoldSim uses them to ensure consistency between inputs and outputs when you link elements. That is, when you define the output attributes for an element, this also affects the required input attributes for that element.

For example, if you assign units of liters to a Reservoir element, GoldSim will only accept an input into the "Initial Value" field for the Reservoir which has
dimensions of volume. (For those readers with programming experience, this is analogous to a programming language which is strongly-typed.)

Read more: Error Checking in Input Fields (page 93).

Note: The strong-typing built into GoldSim may at first appear bothersome: until you get used to ensuring that the inputs to an element are consistent with the specified output attributes, you may wish that GoldSim did not enforce this. It is our experience, however, that users quickly become accustomed to this requirement and greatly appreciate it as they become more proficient with the program. The idea is that a small amount of inconvenience when you first begin to learn the tool is more than made up for in the prevention of errors as you are building your models.

Note: The output attributes refer to the primary output of the element. For elements with multiple outputs, the attributes of the secondary outputs are automatically inferred by GoldSim based on the attributes of the primary output.

Read more: Element Inputs and Outputs (page 74).

For some types of elements (e.g., an Expression), you can specify all three output attributes. For others, some output attributes are fixed and cannot be changed by the user. For example, the output of a Reservoir is always a value (only the order and dimension can be specified), and the output of a logical element (And, Or, Not) is always a scalar condition (in this case, the output attributes are completely fixed).

The output attributes that you will most commonly specify are the dimensions. Dimensions are specified by defining the Display Units for the element. For any element which can be assigned dimensions, the Display Units can be specified on the main dialog (directly below the Description field). Note that if the output is defined as a condition (rather than a value), the Display Units field is grayed out.

Read more: Using Dimensions and Units (page 89).

If the other two attributes (order and type) can be modified for an element, a Type... button will be present in the dialog immediately to the right of the Display Units. Pressing this button displays the following dialog:

The type is specified in the Type box by selecting Value or Condition. The order (scalar, vector, or matrix) is specified by selecting from the Order list. If the order is vector or matrix, you define the Labels for the array in the Row Labels and Column Labels boxes.

Many models can be built using only scalar values. Vectors and matrices, however, are very powerful, and you will want to take advantage of them for more complex simulations.
Creating Elements and Links

Continuous and Discrete Outputs

Read more: Using Vectors and Matrices (page 782).

Although the use of condition outputs is not necessarily required in GoldSim (in many cases you can accomplish the same thing using values), their use can streamline and add clarity to a model. Conditions are particularly useful when modeling discrete events.

Read more: Chapter 5: Simulating Discrete Events (page 351).

In most situations, information is transferred between elements (via links) continuously in time. For example, if Expression element X is defined as being equal to A + B, the values of A and B are continuously sent to X (i.e., the information is "broadcast" through the link throughout the simulation). Conceptually, at any given time in the simulation, X is receiving a signal from A and B.

In order to propagate discrete events (and their consequences) between elements in a model, however, it is necessary to send information between elements intermittently as a "spike" or discrete "packet" of information. To facilitate this, GoldSim allows certain elements to emit and receive (i.e., produce as outputs and/or accept as inputs) discrete signals. Discrete signals are a special category of outputs which emit information discretely, rather than continuously.

Read more: Chapter 5: Simulating Discrete Events (page 351).

Using Dimensions and Units

One of the more powerful features of GoldSim is that it is dimensionally-aware. You enable this capability by assigning display units (and hence dimensions) to the elements (and hence to the inputs and outputs) of your model. GoldSim has an extensive internal database of units and conversion factors. You can enter data and display results in any units. You can even create your own units.

When you create a link, GoldSim ensures dimensional consistency and carries out all unit conversions internally. For example, you could add feet and meters in an expression, and GoldSim would internally carry out the conversion. (If you tried to add feet and seconds, however, GoldSim would issue a warning message and prevent you from doing so.)

You assign dimensions by entering the Display Units. The Display Units determine both the dimensions of the output(s) for the element, and the units in which the output will be displayed (in tool-tips and result tables and charts). You must enter a valid unit string (e.g., m, m^3) made up of unit abbreviations. Appendix D lists all of the internal units (and their valid abbreviations) provided by GoldSim.

The context menu for the Display Units field includes a sub-menu containing valid units and their abbreviations. Clicking on one of these inserts it into the field.
When entering unit strings, the following rules should be followed:

- Units can only be separated by a "-" or a "/". No other operators or blank spaces are allowed. For example, meters per second would be entered as "m/sec". Grams per meter per second would be entered as "g/m-sec" or as "g/m/sec". "g/m sec" would be invalid.

- For clarity, you can use parentheses in unit strings. For example, "kg/(m-sec)" is a valid string.

- Only the first "/" in the string (or the first "/" within a set of parentheses) is recognized, and the rest are ignored. Everything after the slash is assumed to be in the denominator. For example, "1/m/m" and "1/m-m" would both be interpreted as inverse square meters.

- Units are raised to a power by either following the unit with a number or with "^". For example, square meters could be entered as "m2" or "m^2". Inverse seconds could be entered as "sec-1", "sec^-1" or "1/sec". When units are raised to a power, the power must be an integer.

- Metric (SI) units are case-sensitive. For example, Mm and mm represent Megameters and millimeters, respectively.
• If you are referring to a temperature, use C (Celsius), F (Fahrenheit), K (Kelvin), or R (Rankine). If you are referring to a difference in temperature, use Cdeg (Celsius degrees), Fdeg (Fahrenheit degrees), K or R.

• If you are using US units pound or ounce for force or mass, you must distinguish between force and mass by appending an ‘f’ for force (e.g., lbf) and an ‘m’ for mass (e.g., lbm).

• When entering a unit abbreviation in an input field, GoldSim allows you to add an "s" to the end of the abbreviation, and in most cases will recognize it, even if the abbreviation has not been defined. For example, GoldSim will interpret yrs as yr. The rule GoldSim uses is as follows: if GoldSim does not recognize a unit that ends with an "s", it removes the "s" and again tries to recognize it. Note that not all such cases will necessarily work as intended. For example, ms would be interpreted as milliseconds (not meters).

A simple example file illustrating the use of dimensions and units (Units.gsm) can be found in the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

When you enter units within an input field for an element, you can do so by simply inserting them after the values in your expression:

\[3 \text{ m} + 4.5 \text{ ft}\]

You must, however, abide by several rules to ensure that the expression you enter can be unambiguously interpreted by GoldSim:

• Within a unit string, you cannot have any spaces. For example, meters per second must be entered as m/s rather than m / s.

• In general, you should enter a space after each unit string. In some cases, this space is required to ensure that the expression can be unambiguously interpreted (e.g., if the unit string is followed by a subtraction or division). In other cases, even if GoldSim is able to unambiguously interpret the expression, it is good practice to add a space after unit strings to make the expression more readable.

You can also explicitly identify the units by enclosing them in braces:

\[3 \{\text{m}\} + 4.5 \{\text{ft}\}\]

Braces are not required, but they can help make expressions more readable. Moreover, when you use them, it is not necessary to abide by the two rules mentioned above.

**Note:** Braces can only be used in an element's input fields. You cannot use braces when defining the display units for an element.

In general, you only append unit strings to numbers. You should never append a unit string to an output name (a link) which itself has dimensions (this will result in a warning message). You can, however, append units to an output name or expression if the output or expression is dimensionless:
Creating Elements and Links

Dealing with Currency Units

Unlike all other types of units, currency conversion rates cannot be assigned a single constant value (i.e., they change with time). To facilitate using various currency units, GoldSim provides a special dialog for specifying and editing currency units. It is accessed by pressing Model | Currencies… from the main menu.

Note: Specifying and editing currency units is described in detail in Chapter 3 of the GoldSim Financial Module User’s Guide.

Dealing with Temperature Units

You must take special care when using temperature units in GoldSim, since unlike other unit categories, different temperature units are related not just through a conversion factor, but an offset. That is, the zero-values for Celsius (C) and Fahrenheit (F) temperatures are not absolute (i.e., whereas 0 m = 0 ft; it is not true the 0 F = 0 C).

To handle this properly, when you deal with temperatures in GoldSim, you have to be clear whether you are actually specifying a temperature (C, F, or K) or degrees of a particular temperature scale (Cdeg, Fdeg, or K). Temperature and degrees of temperature are treated the same in K, so there is no reason to differentiate when expressing temperature in K.

Use of these units is best explained through examples:

- If the temperature yesterday was 20°C, and today is 5°C warmer, you should calculate today’s temperature as ”20 C + 5 Cdeg”. If you mistakenly entered ”20 C + 5 C”, it would be interpreted as ”293.15 K + 278.15 K” which equals 571.3 K or 298.15°C!

- Similarly, when defining a temperature as a Stochastic, you must be careful. For example, a normal distribution for a temperature defined in Celsius degrees as 20 +/- 5 should be specified as Mean: 20 C, SD: 5 Cdeg. Specifying it incorrectly as Mean: 20 C, SD: 5 C would result in a very large standard deviation!

- If you use a compound unit, you should always specify degree units. For example, when defining of a rate of temperature change of 2.5 Celsius degrees per day, you should enter ”2.5 Cdeg/day”, NOT ”2.5 C/day”.

Automatically Appending Units

When you enter a dimensioned number, link or expression into an input field which expects (based on the element’s display units) a different set of dimensions, the text will be displayed in red, indicating that the input is invalid. For example, if you specify the display units for an Expression element as days, and you define it as ”10 ft”, GoldSim will indicate that there is an error.

Note, however, that if you enter a dimensionless number into an input field that expects a dimensioned input, rather than treating the input as invalid, GoldSim will automatically apply the correct dimensionality to the number when you close the dialog by appending the appropriate units (based on the specified display units for the element). For example, if you specify the display units for an Expression element as days, and you define it as ”10”, GoldSim will automatically append ”days” to the entry when you close the dialog.
Creating Elements and Links

Chapter 3: Building a Model in GoldSim

There are times when you may need to turn a dimensioned value into a dimensionless value in order to use it in an expression. This is necessary, for example, if you need to use an empirical equation (which by definition may be dimensionally inconsistent), or if you need to raise a value to a power, where the power itself has dimensions, such as X^T ime.

GoldSim allows you to do this by casting a dimensioned value or expression to a dimensionless value. For example, the expression

A [ft]

would evaluate A (which, in this case, must have dimensions of length), convert it to a value in feet, and cast it (output it) as a dimensionless value. That is, removes dimensions from a preceding dimensioned value and produces a dimensionless value.

The following are all valid expressions using unit casting:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>A [m^3] + 5</td>
<td>OK if and only if A has dimensions of volume</td>
</tr>
<tr>
<td>(A*B) [ft]</td>
<td>OK if and only if (A*B) has dimensions of length</td>
</tr>
<tr>
<td>Ctime [yr]</td>
<td>OK if and only if Ctime has dimensions of time</td>
</tr>
</tbody>
</table>

A simple example file illustrating casting units (Units.gsm) can be found in the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

As you enter an expression into an input field, GoldSim attempts to interpret the expression, and if it is unable to do so, it indicates this in two ways:

- The item in the expression that is causing the problem is highlighted; and
- A tool-tip is displayed to explain the error.

For example, in the following expression, GoldSim indicates that a right parenthesis is missing by marking as red and underlining the last item in the expression (just prior to the error), and displaying a tool-tip:

\[ A + 5 \]

In this example, GoldSim indicates that a plus sign must be followed by another operand:

\[ A + \]

If you start typing (or move the focus to another field), the tool-tip will disappear. If you stop typing, the tool-tip will reappear.

Note that when evaluating an expression, GoldSim identifies and underlines the first error it encounters. Items shown to the left of the error are displayed in
green (for links), black (for values), gray (for operators) or blue (for units),
indicating that these have been successfully parsed and interpreted. Items
shown to the right of the error are shown in light gray, indicating that they have
not yet been parsed:

\[
\begin{array}{c}
\text{Can't find element 'D'}. \\
A + B + C + D + E
\end{array}
\]

Once you change the focus away from an invalid input field, the entire field will
be highlighted in pink to indicate that there is a problem:

\[
\begin{array}{c}
\text{Equation} \\
A + B + C + D + E
\end{array}
\]

You can view the tool-tip for an input field such as this by simply holding the
cursor over the field (while the focus is elsewhere). The tool-tip disappears
when you move the cursor away.

In some cases, GoldSim may be able to interpret the expression you enter, but
the output attributes may be incorrect. That is, GoldSim uses the specified
output attributes (dimensions, order and type) to ensure consistency when you
enter information into input fields. If the attributes of the specified input
number, link or expression are inconsistent with the required attributes for the
field, the input is considered invalid and the entire field will be highlighted in
pink.

**Read more:** [Understanding Output Attributes](#) (page 87).

For example, if you created a Reservoir element and assigned it display units of
m³, then the Initial Value input for the Reservoir must have dimensions of
volume. If not, the input would be considered invalid:

\[
\begin{array}{c}
\text{Display Units: } m^3 \\
\text{Definition:} \\
\text{Initial Value: } 10 \text{ ft}^3
\end{array}
\]

It is important to understand that a model with an invalid input cannot be run. If
you attempt to run such a model, GoldSim will display an error message.

**Note:** Whenever you change the output attributes for an existing element,
GoldSim will keep the existing input numbers, links or expressions in the input
fields (and mark them as invalid). There is, however, one exception to this rule.
If you change the display units for an element, GoldSim will automatically reset
any inputs which are currently specified as zero to be consistent with the new
dimensions. For example, if you create a new Reservoir element, and assign it
display units of m, and then later change this to gallons (but have not yet defined
an Initial Value), GoldSim will automatically replace the default input for the
Initial Value (0 m) with 0 gal.
Creating Links Using the Link Cursor

The link cursor provides an alternative method for creating links between elements using the browser and input/output interfaces rather than the element's editing dialog.

In particular, double-clicking on an input or an output object in the browser or interface invokes the link cursor. Moving to a compatible (in terms of type, order and dimensions) input or output (in the browser or input/output interface) and double-clicking on it creates a link. If the link is compatible, both "boxes" on the cursor will turn green when you place the link cursor over the input/output to which you wish to link (initially, the bottom box is green and the top box is red).

You can deactivate the link cursor (returning it to a normal cursor) by right-clicking anywhere in the screen or by pressing the Esc key.

A complex model may consist of many hundreds or thousands of individual elements. In order to manage, organize and view such a model it is useful (in fact, essential) to create separate sub-groups or collections of elements.

Such sub-groups are created in GoldSim by placing elements into Containers. A Container is simply a "box" into which other elements have been placed. In a sense, it is like a directory folder on your computer. Containers can be placed inside other Containers, and any number of levels of containment can be created.

The graphics pane in GoldSim shows the elements inside a single container. When you open a file in GoldSim, you will initially be viewing the inside of the Model Container (also referred to as the Model Root). This is analogous to the top-most directory on your computer.

To create a new sub-group of elements within the Model Root (or any other Container) in your model, you need only insert a Container element.

Containers in a model are easily identified in the graphics pane: they all have a small triangle in their upper left-hand corner:

Click the triangle to "enter" a Container.

You can "enter" a particular Container (view its contents) by clicking on the triangle. You can also enter a Container via a Ctrl+double-click on the element (holding the Ctrl key down while double-clicking the left mouse button).

When you create a new Container element, it is initially empty (it contains no elements or objects of any kind). You place new elements in a Container by entering it and inserting elements, or using the context menu of the Container. You can also copy and/or move an element between Containers.

Read more: Moving Elements Between Containers (page 102).

Containers can be placed inside other Containers, and any number of levels of containment can be created. This ability to organize model elements into a hierarchy provides a powerful tool for creating "top-down" models, in which the level of detail increases as you "push down" into the containment hierarchy.

The example below shows a system which has been divided into a number of distinct sub-groups.
The browser shows the containment hierarchy in a manner similar to how a computer's directory hierarchy is shown. You can expand the Container in the browser to show its contents.

You can move through the containment hierarchy using the navigation bar at the top of the graphics pane:

- **Move back to previously viewed Containers**
- **Move forward (after moving back)**
- **Moves up one level**
- **Moves up to top level**

You can also navigate Containers using the following hot-keys:

<table>
<thead>
<tr>
<th>Keys</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alt-Left</td>
<td>Moves back to previously viewed Container.</td>
</tr>
<tr>
<td>Alt-Right</td>
<td>Moves forward (only active if you have already moved back).</td>
</tr>
<tr>
<td>Alt-Home</td>
<td>Moves up to the Model Root.</td>
</tr>
<tr>
<td>Alt-Up or Alt-PgUp</td>
<td>Moves up one level.</td>
</tr>
</tbody>
</table>

Note: Many modern mice come with additional buttons that are typically used to navigate the history of visited “places” (websites or folders). These buttons are also supported by GoldSim. The Backward and Forward buttons go backward or forward in the visited Container history. Pressing either of these buttons while holding the Ctrl key down goes up one level in the hierarchy. Pressing either of these buttons while holding the Ctrl and Shift keys down goes to the top level of the model.
Note: If you hold down the Move Back or Move Forward buttons when you click them, a drop list filled with recently visited Containers (in that direction) will be displayed.

Just to the right of the Container navigation buttons is the Container navigation bar:

Model ▶ Revenues ▶

This shows the full path to the Container you are viewing (in the example above, the “Revenues” Container, which is one level down from the “Model” or highest-level Container is being viewed).

You can use the navigation bar to jump to another Container by clicking on the triangle to the right of any item:

Model ▶ Revenues ▶

“Earnings” and “Expenses” are Containers at the same level as “Revenues” (the currently viewed Container). Clicking on either one of those would jump to that Container.

Model ▶ Revenues ▶

“Consulting”, “Software_Sales” and “Training” are Containers inside “Revenues” (the currently viewed Container). Clicking on any one of these would jump to that Container.

Right-clicking in the Container navigation bar brings up the following context menu:

Model ▶ Revenues ▶

Selecting “Copy path to clipboard” copies the current Container path to the clipboard, allowing you to paste it into other documents (e.g., a report describing the model). Selecting “Browse to path from clipboard” immediately jumps to the path Container path currently stored in the clipboard.

While the various images in the graphics pane can be customized by the model author, the small icons in the browser cannot be changed. Hence, Containers in the browser are always indicated by a small box. Expanding the Container in the browser shows all of its contents (all of the elements it contains).

Read more: Editing the Appearance of Elements (page 434).

Note that the browser and the graphics pane are synchronized. Hence, if you click on an object in the browser, the Container in which the object resides will
Creating Elements and Links

be shown in the graphics pane. This provides a convenient way to navigate through the subsystems in the model.

Read more: Using the Browser (page 105); Using Containers (page 132).

Note: When you save a GoldSim file, GoldSim remembers the last Container that was being viewed, and that Container is shown the next time the file is opened.

Understanding Influences

When you create a link between two elements, GoldSim draws an arrow between the two elements in order to graphically indicate that the two elements are linked.

This arrow is referred to as an influence. If an element in one Container is linked to an element in another Container, an influence will be drawn between the Containers. If multiple elements in one Container are linked to one (or more) elements in another Container, then the influence represents multiple links. Placing your cursor over the link displays a tool-tip indicating the links the influence represents:

Influences have properties, which you can view by double-clicking on the influence arrow (or right-clicking on the influence and selecting Show Links…):
The property dialog for an influence lists all the links represented by the influence. Each link is represented by rows: an Output (where it originates), and an Input (where it ends).

When you select a specific link in the top part of the dialog, it is highlighted in yellow and its details are displayed in the bottom part of the dialog. The details include a listing of the full path of the input or output. The "path" refers to the containment path (e.g., \A\B\C\X indicates that the element named "X" is located in Container C, which is inside Container B, which is inside Container A).

**Note:** Input names are hard-wired. A specific input is referenced in the Influence dialog as ElementId.InputID (e.g., "Volume.Initial Value" refers to the Initial Value input of the Reservoir element named Volume).

**Read more:** [Element Inputs and Outputs](#) (page 74).

You can jump to the element associated with a particular input or output by clicking on the path in the Link Details section.

The details also describe the Output Type (e.g., value, condition), Units, and Order (e.g., Scalar, Vector or Matrix). The Link Type is also indicated. Most links in a model will be “normal” links. Other types of links (e.g., coupled links and Previous-value links) can only be created by using one of the extension modules to GoldSim (e.g., the Contaminant Transport Module).

By default, all influences are curves (drawn with no curvature). You can add curvature to the influence by selecting it and dragging the control point (the small square) with your cursor:

You can modify the appearance of influences by changing their color and thickness, changing their shape (straight lines or orthogonal lines), adding nodes to segment straight lines, and adding text.

In some instances, in order to simplify the appearance of the graphics pane you may want to hide all the influences of a particular type (e.g., hide all influences consisting of condition outputs). This is referred to as filtering the influences.

**Read more:** [Editing the Appearance of Influences](#) (page 426); [Filtering Influences](#) (page 430).

**Note:** Influences are not drawn between the element(s) being referenced by a Result element (a specialized element in GoldSim) and the Result element itself. This is because unlike other elements in GoldSim, when you reference an output of an element within a Result element, it is not treated as a “link”. GoldSim provides other tools for viewing these kinds of connections.

**Read more:** [Browsing Between Result Elements and Referenced Outputs](#) (page 574).

When an element is linked to an element in another Container, it is useful to be able to visually signify this (i.e., since the linked element is in a different
Container, when viewing the element there is no port to which the influence can be connected). GoldSim provides a solution to this problem. If an element is linked to an element in a different Container, GoldSim indicates this by placing a dot in the input and/or output port:

![Diagram showing output of one element connected to input of another](image)

**Read more:** Understanding Input and Output Ports (page 76).

Right-clicking an outlined port accesses a context menu with the option to “Show Links from Element” (for output ports) or “Show Links to Element” (for input ports). Selecting this option will open a dialog showing all the links from or to the element, respectively:

![Link dialog](image)

Each link is represented by rows: an Output (where it originates), and an Input (where it ends).

When you select a specific link in the top part of the dialog, it is highlighted in yellow and its details are displayed in the bottom part of the dialog. The details include a listing of the full path of the input or output. The "path" refers to the containment path (e.g., \Container2\Flow indicates that the element named "Flow" is located in a Container named Container2).

**Note:** Input names are hard-wired. A specific input is referenced in the Influence dialog as ElementId.InputID (e.g., "Lake.Addition_Rate" refers to the Addition Rate input of the Reservoir element named Lake).

**Read more:** Element Inputs and Outputs (page 74).
Creating Elements and Links

You can jump to the element associated with a particular input or output by clicking on the path in the Link Details section.

The details also describe the Output Type (e.g., value, condition), Units, and Order (e.g., Scalar, Vector or Matrix). The Link Type is also indicated. Most links in a model will be “normal” links. Other types of links (e.g., coupled links and Previous-value links) can only be created by using one of the extension modules to GoldSim (e.g., the Contaminant Transport Module).

In a dynamic simulation, it is often necessary for you to explicitly reference time in a mathematical expression that you enter as an input to an element (i.e., you might want an input to take on different values at different times).

Within GoldSim, you can reference the elapsed time since the simulation began as \( ETime \). For example, the input to the following Expression element takes on a value of 10 for times less than 1 year, and 20 afterward:

$$E(\begin{array}{c} 
ETime < 1 \text{ yr} \text{ then } 10 \\
\text{else } 20 
\end{array})$$

The if function is one of the built-in functions provided by GoldSim. This function has three arguments and can be written in two ways:

$$\text{If}(A \text{ then } B \text{ else } C) \text{ or } \text{If}(A, B, C)$$

The above would be interpreted as "If A is a true condition, the expression is equal to B, otherwise it is equal to C".

Read more: Entering Mathematical Expressions into Element Input Fields (page 124).

GoldSim also allows you to begin your simulation at a specific date/time (e.g., January 1, 2005) and reference the calendar (as opposed to elapsed) simulation time (as \( \text{DateTime} \)):

$$\text{If}(\begin{array}{c} 
\text{DateTime} > "15 \text{ April } 2009" \text{, } 10 \\
\text{else } 20 
\end{array})$$

Read more: Referencing Dates in Expressions (page 131).

Both ETime and DateTime are actually special model outputs referred to as Run Properties.

Read more: Understanding and Referencing Run Properties (page 486).

Note: When you place the cursor over an element, an input or an output prior to running a simulation, GoldSim displays a tool-tip showing the object's current (expected) value. If the object is a function of ETime, by default this calculation assumes a value of 0s. You can modify the value used in such a calculation by editing Time to use for Edit Mode updates in the Advanced Time Settings dialog (accessed from the Advanced… button on the Time tab of the Simulation Settings dialog).

Read more: Using Tool-Tips (page 109); Referencing Time in Edit Mode (page 478).

You can cut, copy and paste elements to and from the Windows clipboard using the Ctrl+X, Ctrl+C and Ctrl+V keys, respectively. Elements can be deleted by selecting the element and pressing the Delete key. These actions can also be
carried out from the main menu (under **Edit**) or from the context menus for elements and the graphics pane.

Multiple elements can be selected, and then cut or copied to the clipboard and pasted. You can also cut or copy an entire Container, in which case the Container and all of its contents will be copied to the clipboard.

---

**Warning:** When you paste an element, GoldSim will append a number to the name if it conflicts with an element at the location where it is being pasted. When you paste an entire Container, GoldSim will automatically localize the Container if the names of elements inside the pasted Container conflict with those at the location where it is being pasted. Elements with the same name will conflict if they are located in the same scope. Localizing a Container changes the scope of its contents.

---

**Moving Elements Between Containers**

Read more: [Localizing Containers](#) (page 944).

Often while building a model you will want to reorganize your model by adding Containers and moving elements into them, or moving elements between existing Containers. GoldSim provides two methods by which you can move one or more elements between Containers.

For both methods, you must first select the element (or elements) which you wish to move (multiple elements can be simultaneously selected).

---

**Note:** You can use these same methods to move graphical objects (such as images and text) that you have added to the graphics pane. You can also simultaneously select and move elements and graphical objects from one Container to another.

---

The first method for moving the elements(s) involves using the context menu for the object. After selecting the elements(s), access the context menu for a selected element (via a right-click) and choose **Move To...**. A browser window showing all of the Containers in the model will be displayed.
Select a Container and press **OK** to move the objects(s).

Alternatively, you can drag the selected element(s) within the graphics pane to a Container present in the graphics pane. When you do this, the Container will be outlined in a blue box. When you then release the left mouse button, the element(s) will be moved into the Container. (Note that you cannot drag elements out of the graphics pane onto the browser or onto a different model.)

**Note:** If you try to move an element into a localized Container such that a naming conflict occurs (i.e., because the Container already has an element with that name inside), GoldSim will warn you and provide you with an opportunity to change the name of the element being moved.

**Warning:** Whenever you paste or move an element, GoldSim automatically updates all input expressions in the model and rebuilds the **causality sequence** to ensure that the pasted/moved element is properly linked to other elements in the model. For large models (many hundreds of elements) this process could take a noticeable amount of time. You can disable this automatic updating by clearing the "Update expressions automatically after all moves and pastes" checkbox in the Options dialog (accessed via **Model | Options**…). If the option is cleared, you can still manually force an update by pressing F9 or selecting **Model | Update Expressions**. Regardless of whether this option is checked, however, GoldSim always carries out an update prior to running a model.

**Read more:** [Localizing Containers](#) (page 944).

**Read more:** [The Causality Sequence and Element Updating](#) (page 343).

In many cases, you may wish to create different portions of your model in different model files. For example, one individual may be working on one part of a model (a particular subsystem), while another is working on a different subsystem. At some point, you would then want to combine the various subsystems into a single model file.
Navigating and Viewing a Model

GoldSim allows you do so by simply opening two instances of GoldSim (each with a different file), copying the desired elements (including entire Containers) to the clipboard, and then pasting them into the other model. When doing so, however, you must be aware of three points:

- Multiple objects can be copied to the clipboard at one time (e.g., you can select multiple objects and copy them to the clipboard). You can also copy an entire Container (and all of its contents) to the clipboard.
- When copying Containers between models, GoldSim will localize the Container if the names of elements inside the pasted Container conflict with those in the model into which it is being pasted.
- If you are copying elements with vector or matrix inputs or outputs between models, you must ensure that the Array Labels defining those vectors and matrices are identical in both models. If they are different, it would cause a conflict when the item is pasted, and GoldSim will issue a warning message and prevent you from doing so.

Read more: Localizing Containers (page 944); Using Vectors and Matrices (page 782).

Navigating and Viewing a Model

GoldSim's graphical user interface provides a number of powerful tools to make it easy for you to navigate and view both simple and complex models. While the graphics pane represents the "canvas" on which you graphically build a GoldSim model, the browser represents the "road map" allowing you to quickly navigate through your model. Tool-tips within both the graphics pane and the browser allow you to quickly obtain an overview of a model without constantly opening property dialogs. Powerful search capabilities allow you to find any element in your model. And perhaps most importantly, the ability to easily view model dependencies (who affects who?) allows you to better understand and interpret the behavior of complex models.

Each of these features is discussed in detail below.

The graphics pane shows the contents of a single Container. You can move around within the graphics pane using the vertical and horizontal scroll bars and scroll arrows at the right and at the bottom of the pane.

Read more: Understanding Containers (page 95).

Note that if you scroll far enough horizontally or vertically you will eventually reach the "edge" of the graphics pane. That is, the graphics pane represents a document of fixed size.

Read more: Adjusting the Size of the Graphics Pane (page 425).

You can change the scale at which the graphics pane is viewed (i.e., zoom in and zoom out). There are two ways to do this:

- In the lower right-hand corner of the screen (the right side of the status bar) is a zoom slider:

  ![Zoom Slider](image)

  By dragging the slider (left or right) you can adjust the scale. Clicking on the "-" or "+" increments the scale up and down in 10% increments.
If your mouse has a wheel, you can also use this for zooming. Rolling the wheel toward you while holding down the Ctrl key zooms out. Rolling the wheel away from you while holding down the Ctrl key zooms in.

- Clicking the **Current scale** (100% in the example above) opens a zoom dialog:

![Zoom Dialog](image)

Selecting an option adjusts the zoom level accordingly.

**Note:** By default, GoldSim remembers the position and scale of a Container when you leave it, so that when you return to it, it is displayed at the same scale and in the same position. You can disable this feature via a checkbox in the **Graphic** tab of the Options dialog.

### Using the Browser

For simple models containing a small number of elements with only a few subsystems (Containers), you may wish to only use the graphics pane to navigate and view your model. For most models, however, use of the browser becomes essential.

By default, the browser is activated and is initially docked on the left side of the screen:
The browser can be in four different states (Floating, Docking, Auto Hide and Hide), which can be selected from the drop-list at the top of the browser:

You can make the browser become a floating window by selecting **Floating** from the browser drop-list:

You can dock a floating browser (to any side of the screen) by starting to drag the browser. When you do so, multiple docking points appear on the screen:
Navigating and Viewing a Model

Dragging to the docking point docks the browser at that location.

You can hide the browser completely by selecting **Hide** from the browser drop-list. You can also hide the browser by pressing **F6** or by pressing the browser button in the Standard toolbar:

The final option is to **Auto Hide** the browser. If you select this option, the browser is hidden until you move the cursor over the vertical section marked “Browser” on the left side of the screen:

When you do so, the browser opens (and stays open until ther cursor moves outside of the browser).

Hiding (or auto hiding) the browser is useful, particularly for complex models, as it allows more of the graphics pane to be visible.

The browser organizes the model into one of two views in a tree structure, which greatly facilitates navigation of complex models. The two browser views are referred to as **Containment View** and **Class View**, and you toggle between these alternative views using the tabs at the bottom of the browser.
Navigating and Viewing a Model

The default browser view is Containment View. In Containment View, elements are organized in a hierarchical manner (by containment), similar to the way that files and directories on a computer are organized.

In this view, elements at the same hierarchical level are listed in alphabetical order (except for Containers, which are always listed first).

Read more: Understanding Containers (page 95).

In Class View, rather than being organized by containment, the browser is organized by element type.

Expanding a folder shows all of the elements of that type.

All the elements of a particular type are listed together (alphabetically) in a folder labeled with the element type. This provides a very convenient way to find elements of a specific type. By default, when in Class View, the last three Containers of the path for the element is shown in parentheses next to the element ID. You can hide this path by deselecting Show Path in the context menu for the browser.

The browser and the graphics pane are synchronized. This means that if you click on an object in the browser, the Container in which the object resides will be opened and its contents will be shown in the graphics pane. Similarly, if you
click on an element in the graphics pane, the Container in which the element resides will be expanded and that element will be selected in the browser. This synchronization facilitates navigation through the subsystems in your model.

By default, GoldSim will only display the element names in the browser. If you select Show Element Subitems from the context menu for the browser, press Ctrl+I, or press View | Show Element Subitems from the main menu, the element inputs and outputs will also be displayed:

![Browser Screenshot]

When you expand an element within the browser, inputs are listed first (and represented by an arrow head pointing to the left). Outputs are listed last (and represented by an arrow head pointing to the right).

The inputs and outputs are color-coded:

- Inputs/outputs representing values are blue.
- Inputs/outputs representing conditions are green.
- Inputs/outputs representing discrete signals (transactions) are red.
- Inputs/outputs representing complex outputs (e.g., Time Series definitions) are gray.

In addition,

- An input which contains a link to another element has a dot immediately to the left of the arrow head.
- An output which is linked to another element has a dot immediately to the right of the arrow head.
- An output that is a state variable has a bold vertical line to the left of the arrow head.

Read more: Understanding State Variables in GoldSim (page 341).

**Using Tool-Tips**

Elements, input objects and output objects, and input fields all display tool-tips when the cursor is placed over them.

**Element Tool-Tips**

Holding your cursor over an element in the graphics pane (or the browser) displays a tool-tip:
Element tool-tips display three types of information:

- Basic information
- State
- Properties

All elements display basic information regarding the element consisting of the element’s name, an icon (identifying the element type), and the Description (if one is specified).

In addition to this information, other basic information about the element may also be displayed, depending on the element type, as well as how it has been defined. If the element has a primary output, the tool-tip will show the display units for the output (in parentheses). If the element is a vector or a matrix, the array label set(s) will be displayed (in brackets).

Read more: Element Inputs and Outputs (page 74); Using Vectors and Matrices (page 782).

To augment this basic information, State and Property information is also displayed for some elements. When displayed, these are separated from the basic information by a line.

State information consists of a current or initial Value for the element. Only elements that have a primary output display State information. GoldSim has three primary modes (or states) that a model can be in: Edit Mode, Run Mode and Result Mode. When in Edit Mode (prior to running the model), the current or initial value is shown in the tool-tip. This represents the value of the output prior to running the simulation.

Read more: Simulation Modes (page 66).

If an element has invalid inputs such that a current value cannot be computed, the value is reported as "undefined".

When the model has finished and has results (Result Mode), the Value shown in a tool-tip represent is the final value (i.e., the value at the end) of the final realization.

While a model is running (Run Mode), it can be paused and the model can be navigated. In this case, the Value shown in a tool-tip represents the last value that was calculated (since the simulation has not yet been completed).

In addition to the basic information and the State (value) information, some elements will also display some of their input properties in the tool-tip. This allows you to view the key inputs for the element without having to open the dialog.

Note: You can control the number of significant figures displayed in tool-tips from the Results tab of the Options dialog (accessed via Model | Options… from the main menu).
Input Field Tool-Tips

Read more: The Results Tab of the Options Dialog (page 448).

Holding your cursor over or placing your cursor into an input field in a dialog displays a tool-tip showing the computed value of the entry or expression in the input field:

![Current Value = 1685 m3/day](image)

The value is computed based on the current value when in Edit Mode and the last value when in Result Mode (the final value of the final realization) and Run Mode (the last value calculated).

If you place your cursor on a link from another element and left-click, that link will be selected (and highlighted in italics), and the tool-tip will display the value just for that link (as opposed to the entire input field):

![Flow1 + Flow2](image)

These kinds of tool-tips can be very useful when debugging and/or exploring models. Note that the icon for the link is also provided (identifying the element type).

Information regarding GoldSim’s built-in constants is also displayed in expressions via tool-tips:

![c (m/s) Speed of light](image)

If the input field is invalid for some reason, the tool-tip will display an appropriate error message:

![Missing right parenthesis.](image)

If you start typing (or move the focus to another field), the tool-tip will disappear. If you stop typing, the tool-tip will reappear.

Read more: Error Checking in Input Fields (page 93).

Note: You can control the number of significant figures displayed in tool-tips from the Results tab of the Options dialog (accessed via Model | Options… from the main menu).

Read more: The Results Tab of the Options Dialog (page 448).
GoldSim carries out all of its internal calculations using double precision numbers. In general, this means that GoldSim can internally handle absolute values as small as $2.2 \times 10^{-308}$ and as large as $1.8 \times 10^{308}$. If you enter (or GoldSim calculates) a value whose absolute value is outside this range, the tool-tip for the element and the output will be displayed as $1.\#INF$ (or $-1.\#INF$).

**Note:** Although internal calculations are carried out using double precision numbers, results are only stored as single precision numbers (in order to reduce storage requirements). This means that when results are viewed in tables or charts (see Chapter 8), the range of absolute values that can be displayed is between $1.2 \times 10^{-38}$ and $3.4 \times 10^{38}$.

**Finding Elements**

To assist in finding specific elements in a model, GoldSim provides a Search utility.

To start a search, press the Search button:

![Search button](image)

When you do so, the following dialog will be displayed:

![Search dialog](image)

(You can also open this dialog by pressing Ctrl+F).

Enter a string and press the **Find Next** button (or press **F3**) to find the next item in the browser that matches the search string. This will close the dialog and carry out the search. The search parameters are saved, so you can (repeatedly) press **F3** to find the next item in the browser that matches the search string.

Once you have moved through the browser tree to more than one item, you can press **Shift-F3** to jump to back to to a previous item (as opposed to the next item).

The search is not case-sensitive.

You can control where GoldSim searches selecting whether you want to search in IDs, Descriptions, and/or Notes.

*Read more:* [Creating, Editing and Viewing Notes](page 755).

**Note:** These Search options are stored in the Windows Registry (so that once you change them, they will remain until you edit them again, even when editing other GoldSim files).

**Viewing Element Dependencies**

In complex models, it is often useful to explore the interdependencies of the various elements (i.e., who affects who). GoldSim provides two very powerful utilities for doing this: the **Function Of View**, and the **Affects View**. If you right-click on an element in either the graphics pane or the browser (to access the context menu) and select **Function Of…,** a floating window is displayed:
This is a specialized browser view of the model. It starts with the selected element, and only shows those elements which affect that element (i.e., those elements which the selected element is a function of). In the above example, “Pond2” is a function of one element: “Route_Overflow”. “Route_Overflow”, in turn, is a function of “Fraction_to_Pond2” and “Pond1”. “Pond1” is a function of “Inflow”, “Initial_Volume” and “Pond_Capacity”.

Similarly, if you select Affects... from the context menu, a window like this will be displayed:

This view of the model starts with the selected element, and only shows those elements which are a function of that element (i.e., those elements which the selected element affects).

In the above example, “Inflow” affects “Pond1”, which affects “Route_Overflow”, “Pond2” and “Pond3”.

>Note: The Function Of... and Affects... options are only available if the element is either a function of other elements or affects other elements.

These two browser views have all the functionality of the regular browser (e.g., tool-tips, context menus). If you double-click on an element in one of these views, its property dialog is displayed.

In addition, the context menu for an item in the Function Of or Affects browser includes an option to immediately jump to the opposite view. That is, if you are viewing a Function Of list, you can right click on any element in the list, and immediately jump to the Affects list for the selected element.

By default, Function Of and Affects browser views are synchronized with the main browser (and hence the graphics pane). As a result, as you explore the interdependencies of the elements, you can simultaneously observe where they are located in the containment hierarchy.
Note: If desired, you can turn off the synchronization by right-clicking on white space in the Function Of or Affects view (i.e., not on an element), and clearing the Synchronize option in the context menu.

Note that if you have defined a looping model, GoldSim will stop at the point of recursion. For example, if “Pond” is a function of “Evaporation”, and “Evaporation” is a function of “Pond”, the Function Of view for “Pond” would look like this:

Read more: Evaluating Feedback Loops (page 346).

If an element is a function of a locally-available property (such as a Run Property), the Function Of view will indicate that it is a function of the Container that “owns” the property (typically the Model Container):

In this example, Inflow_Rate is a function of ETime (a Run Property)

Read more: Understanding and Referencing Run Properties (page 486).

Note: Result elements do not appear in the “Affects” list for elements that they reference. This is because unlike other elements in GoldSim, when you reference an output of an element within a Result element, it is not treated as a “link”. GoldSim provides other tools for navigating these kinds of connections.

Read more: Browsing Between Result Elements and Referenced Outputs (page 574).

The Function Of and Affects have a search field at the top. Enter a string and press the Search button to the left of the string (the magnifying glass) or press F3 to find the next item in the list that matches the search string. The search is not case-sensitive.

You can control where GoldSim searches by pressing Search Options..., which will display a dialog for selecting whether you want to search in IDs, Descriptions, and/or Notes.
Running a Model and Viewing Results

This section provides a very brief overview of running a model and viewing results. These topics are discussed in detail in Chapters 7 and 8. The objective of this section is to provide you with enough basic information to run and view the results of simple models.

GoldSim is a dynamic simulator, which means that your model can evolve and change with time. In order to carry out a dynamic simulation, GoldSim steps through time in discrete intervals (referred to as timesteps).

Calculations are carried out every timestep. In GoldSim, you specify the duration of the simulation (e.g., 1 year) and the length of timestep to use (e.g., 1 day).

The appropriate timestep length is a function of how rapidly the system represented by your model is changing: the more rapidly it is changing, the shorter the timestep required to accurately model the system. GoldSim allows you to change the timestep during a simulation (e.g., use short timesteps at early times when things are changing rapidly, and larger timesteps at later times).

To run a model, you must first specify its simulation settings. When you create a new model, the simulation settings dialog is automatically displayed. After the model has been created, the simulation settings can be accessed and edited from the main menu under Run | Simulation Settings, or by double-clicking on the Simulation Settings button in the standard toolbar. When you do so, the following dialog is displayed:
Running a Model and Viewing Results

The **Time** tab of the dialog is used to specify the duration of the simulation, the timestep length, and the units in which time will be displayed in results.

The **Monte Carlo** tab of the dialog is used to specify Monte Carlo options (how probabilistic simulations will be carried out), such as the number of realizations, and whether Latin Hypercube sampling is to be used.

**Note:** Appendix A provides an introduction to probabilistic simulation techniques, and introduces the basic terminology required to understand the Monte Carlo options presented in this dialog.

---

**Saving Results**

**Read more:** [Simulation Settings](page 452).

**Before you run a model, you select the results that you wish to save. With regard to saving results, there are two things that you must specify:**

- *Which* specific results do you want to save (i.e., which outputs of which elements)?
- *When* do you want to save these results?

**Within GoldSim, there are two fundamental types of results:**

**Final Value Results:** These are usually used to view results involving multiple realizations (or items in an array), and allow you to display distributions, multi-variate results (e.g., scatter plots) and array plots. By default, “Final Value” type results are only available at the final time point in the simulation. In some cases, however, you may also want to view distributions or scatter plots at other points in time (other than the end of the simulation). GoldSim facilitates this by allowing you to create Capture Points at which these results are also made available.
Running a Model and Viewing Results

**Read more:** Creating Capture Points for Final Value Results (page 469).

**Time History Results:** These are saved at selected timesteps, and allow you to display how an output changes as a function of time.

All element property dialogs have check boxes (at the bottom of the dialog) to specify whether output(s) of the element are to be saved. You can save the **Final Values** (the values at the end of each realization in the simulation) and/or **Time History** (the value at selected timesteps throughout the simulation).

By default, when you create a new scalar element, the Save Results checkboxes will be checked (on). For array elements, the **Time History** checkbox defaults off.

The **Final Values** checkbox always controls whether Final Value results are saved. However, the **Time History** checkbox can be overridden. In particular, it is always applied for single realization runs, but is overridden for multiple realization runs. In particular, Time History results for multiple realization runs are only saved for outputs that are connected to Result elements.

**Read more:** Saving Outputs as Results (page 494).

When Time History results are saved, by default GoldSim saves the values of outputs at every Basic Step (and Reporting Period). You can instruct GoldSim to only save results at selected timesteps (e.g., every tenth Basic Step) when defining timestepping options in the Simulation Settings dialog.

**Read more:** Understanding Timestepping in GoldSim (page 455); Specifying When Time History Results Will Be Saved (page 464).

**Running a Model**

Selecting Run | Run Model, pressing F5, or pressing the Run button in the Run Control toolbar causes GoldSim to check the model for errors (to see if it can be run).

![Run button](image)

*The second button from the left in the Run Control toolbar is the Run button.*

If an error is found in the model such that it cannot be run, GoldSim will display a fatal error message describing the problem. In most cases, a button on the dialog displaying the error message will allow you to jump directly to the element causing the problem.

If GoldSim does not detect any errors, the model will be placed into Run Mode and the simulation will begin. While the model is running, the status bar will display the status of the run (i.e., current realization, timestep, and elapsed time).

In addition to running the model, the Run Control toolbar can also be used to pause and resume a simulation, and/or step through the simulation one timestep or one realization at a time.

**Read more:** Running and Viewing the Status of a Simulation (page 497).

A small dialog will alert you when the simulation is complete:
Clicking the **OK** button in this dialog (the only button) will place the model into Result Mode.

If there were any warning messages generated during the simulation, you will immediately be given an opportunity to view the **Run Log**. The Run Log contains basic statistics regarding the simulation (e.g., the version of the program, the date, the simulation length), and any warning or error messages that were generated.

**Read more:** [The Run Log](#) (page 548).

After a simulation is complete, the model will be placed in Result Mode. This will be indicated by the status bar (“Results” will be displayed on the left side of the status bar and the status bar will be green):

More importantly, editing the model’s logic (i.e., the properties of model objects) will be disabled. In addition, the cursor takes on a different appearance (it turns green) to indicate that editing is disabled.

**Note:** In order to edit the model again, you need to return to Edit Mode. You can do so by pressing **F4** or the Edit Mode button:

In Result Mode, an output’s Last Value (the value at the end of the last realization) will be displayed in a tool-tip when the cursor is held over it in a browser (or output interface). If the output is the primary output of an element, the Last Value will also be shown in the tool-tip displayed when the cursor is held over the element itself. The Last Value is displayed in tool-tips regardless of whether results (Time Histories or Final Values) have been saved for the output.

When in Result Mode, elements which have saved results will be identified in two ways: 1) the elements (and their outputs) will be **bold** in the browser; and 2) the element output ports in the graphics pane will be green:

**Read more:** [Understanding Input and Output Ports](#) (page 76).

Right-clicking on an element with a primary output (in the graphics pane or browser) or on a specific output (in an output interface or a browser) which has been saved will provide a context menu for displaying results.
Depending on the type of output and which results have been saved, up to three different options will be displayed at the top of the menu.

Clicking on one of the items at the top of this menu displays the results for the selected object.

Read more: Element Inputs and Outputs (page 74).

This manner of viewing results in GoldSim is so fundamental to the GoldSim design that it bears repeating: To view the result of an element, right-click on the element (or one of its outputs) and select the appropriate result type at the top of the menu.

Note: Time history results may not be available in this manner, depending on the nature of the simulation. In particular, for runs with multiple realizations, extra steps must be taken to save and view time history results.

Read more: Chapter 8: Displaying Results in GoldSim (page 555).

We close this chapter by walking through a simple example to allow you (if you have not already done so) to build and run your first GoldSim model.

This simple example uses a Reservoir element to simulate the volume of water in a pond. The completed model file (FirstModel.gsm) can be found in the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

1. Open GoldSim.
2. Select New Model from the Start dialog.
3. Open the Simulation Settings by pressing F2.
4. In the Time tab of the Simulation Settings dialog, you will note that “Elapsed Time” is selected. Leave this unchanged.
5. Set the Duration to 20 days.
6. Set the Basic Step to 0.2 days.
7. Press OK to close the dialog.
8. Insert a Data element using the context-sensitive menu of the graphics pane (right-click in the graphics pane, select Insert Element, go to the Inputs category, and click on Data).
9. The element will be named Data1 by default. Change the name to Initial_Volume and give it a Description (Initial volume of water in the pond).
10. Specify the Display Units as cubic meters (m3).
11. Specify the entry in the Data Definition input field as 10 m3. The dialog should look like this:
12. Press **OK** to close the dialog.

13. Insert an Expression element at a point below the Data element using the context menu of the graphics pane (right-click in the graphics pane, select Insert Element, go to the Functions category, and click on Expression).

14. The element will be named Expression1 by default. Change the name to Inflow_Rate and give it a Description (The inflow rate varies with time).

15. Specify the Display Units as cubic meters per day (m3/day).

16. We will make the expression a function of time, utilizing GoldSim's *if..then* function. In particular, we will specify that for the first 10 days, the rate is equal to 5 m3/day, and it is equal to 2 m3/day afterward. To do this, enter the following into the Equation field for the Expression:

17. Press **OK** to close the dialog.
18. Insert a Reservoir element to the right of the Data and Expression elements using the context menu of the graphics pane (right-click in the graphics pane, select Insert Element, go to the Stocks category, and click on Reservoir).

19. The element will be named Reservoir1 by default. Change the name to Volume and give it a Description (Volume of water in pond).

20. Specify the Display Units as cubic meters (m³).

21. For the Initial Value input, type Initial_Volume. In the Additions section, specify the Rate of Change as Inflow_Rate. The dialog will then look like this:

![Reservoir Properties: Volume dialog]

22. Press OK. The graphics pane should look similar to this:

![Initial_Volume and Volume elements]

If it doesn't, you can drag the elements around the graphics pane so that it does.

23. Save the file by pressing the Save button on the main menu. You will be prompted for a filename.

24. We are now ready to run the model. To do this press the Run button in the Run control toolbar (or press F5).
25. You will be presented with a dialog stating that the simulation has completed. Press OK to close the dialog.

26. You can now view results. To do so, right-click on Volume (the Reservoir element) and select **Time History Result…**. The following result will be shown:

![Time History Result](image)

Congratulations! You have built and run your first GoldSim model. The details of using the various elements, customizing the interface, running the model, viewing results and documenting your model are provided in the remaining chapters. You should certainly have enough information at this point, however, to begin to experiment with GoldSim.
Chapter 4: Using the GoldSim Elements

“The time has come,” the Walrus said,
“To talk of many things:
Of shoes, and ships, and sealing wax
Of cabbages and kings
And why the sea is boiling hot
And whether pigs have wings.”

Lewis Carroll

Chapter Overview

This chapter describes the details of using each of the basic GoldSim elements. After first describing the details of a topic that is applicable to all elements (entering expressions into input fields), the manner in which each of the basic GoldSim elements is defined and used is presented. A number of advanced elements are described in Chapter 10.

In this Chapter

The following topics are discussed in this chapter:

• Entering Mathematical Expressions into Element Input Fields
• Using Containers
• Overview of GoldSim Element Types
• Using Basic Input Elements
• Using Time Series Elements
• Using Stock Elements
• Using Basic Function Elements
• Using Delay Elements
• How GoldSim Carries Out its Calculations
Entering Mathematical Expressions into Element Input Fields

GoldSim provides a number of features that enable you to enter mathematical expressions into input fields. This section provides information on the use of these features. This includes a discussion of GoldSim's built-in functions and constants, and guidance with regard to creating conditional (logical) expressions and using dates in expressions.

GoldSim provides a wide variety of built-in functions that can be used in expressions. These functions can be typed in directly, or inserted using the context menu of an input field:

The functions are divided into five categories:

1. Trigonometry functions, such as sine and cosine;
2. Math functions, such as square root and logarithm;
3. Special functions, such as the Bessel function and "if,then,else";
4. Financial functions, such as present value and future value; and
5. Array operators, which are specialized functions that operate only on vectors and matrices.

An example of the use of a built-in function is shown below:

\[ \text{Equation} \]

\[
\text{max}(a,b)
\]

The "max" function, shown above in the Equation field, returns the maximum of a list of arguments.

The arguments to a built-in function are enclosed in parentheses, and if there are multiple arguments, they are separated by commas. The arguments to the built-in functions can be links, constants or expressions, but typically must meet certain requirements (e.g., the arguments to a "max" must all share the same dimensions). The first four categories of functions (and any requirements imposed on their arguments) are summarized below.

**Trigonometry Functions**

GoldSim provides the following trigonometry functions:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sin(A)</td>
<td>Sine of A</td>
</tr>
<tr>
<td>cos(A)</td>
<td>Cosine of A</td>
</tr>
<tr>
<td>tan(A)</td>
<td>Tangent of A</td>
</tr>
</tbody>
</table>
### Entering Mathematical Expressions into Element Input Fields

**Math Functions**

GoldSim provides the following standard mathematical functions:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cot(A)</td>
<td>Cotangent of A</td>
</tr>
<tr>
<td>sinh(A)</td>
<td>Hyperbolic sine of A</td>
</tr>
<tr>
<td>cosh(A)</td>
<td>Hyperbolic cosine of A</td>
</tr>
<tr>
<td>tanh(A)</td>
<td>Hyperbolic tangent of A</td>
</tr>
<tr>
<td>asin(U)</td>
<td>Arcsine of U. The result has dimensions of an angle.</td>
</tr>
<tr>
<td>acos(U)</td>
<td>Arccosine of U. The result has dimensions of an angle.</td>
</tr>
<tr>
<td>atan(U)</td>
<td>Arctangent of U. The result has dimensions of an angle.</td>
</tr>
</tbody>
</table>

A: An angle (i.e., a value). It can have angular units or no units, in which case it is assumed to be in radians. Can be a scalar or an array.

U: Must be a unitless value. Can be a scalar or an array.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs(X)</td>
<td>Absolute value of X</td>
</tr>
<tr>
<td>exp(U)</td>
<td>Exponential (e^U)</td>
</tr>
<tr>
<td>ln(U)</td>
<td>Natural logarithm of U</td>
</tr>
<tr>
<td>log(U)</td>
<td>Logarithm base 10 of U</td>
</tr>
<tr>
<td>max(X,Y,...)</td>
<td>Returns the maximum of a list of arguments. The first argument determines the order of the output. Arguments must either be all scalars, all arrays of the same order (and set of array labels), or a mixture of arrays of the same order and scalars (in which case the first argument cannot be a scalar).</td>
</tr>
<tr>
<td>min(X,Y,...)</td>
<td>Returns the minimum of a list of arguments. The first argument determines the order of the output. Arguments must either be all scalars, all arrays of the same order (and set of array labels), or a mixture of arrays of the same order and scalars (in which case the first argument cannot be a scalar).</td>
</tr>
<tr>
<td>mod(X,Y)</td>
<td>Modulus: remainder of X/Y; sign is sign of X. The first argument determines the order of the output. Arguments can be scalars, arrays of the same order (and set of array labels) or mixed (in particular, if X is an array, Y can be a scalar; if X is a scalar, however, Y cannot be an array). If X and Y have dimensions, they must have the same dimension (although the units can be different), and the result has the same dimension as X and Y.</td>
</tr>
<tr>
<td>round(U)</td>
<td>Rounds off U</td>
</tr>
<tr>
<td>sqrt(X)</td>
<td>Square root of X. Note: if X has units, they must be an even power, for example: sqrt(5 m^2) is OK, but sqrt(5m) is not.</td>
</tr>
<tr>
<td>trunc(U)</td>
<td>Truncates U</td>
</tr>
<tr>
<td>ceil(U)</td>
<td>Returns the next integer that is greater than U</td>
</tr>
<tr>
<td>floor(U)</td>
<td>Returns the next integer that is less than U</td>
</tr>
</tbody>
</table>

U: Must be a unitless value. Can be a scalar or an array.

X, Y: A value. Can be a scalar or an array.
GoldSim allows for a fractional error of 1e-12 when evaluating the ceil() and floor() functions. For example, although ceil(4.000001) returns 5, ceil(4.000000000001) returns 4.

**Special Functions**

GoldSim provides the following specialized mathematical functions:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>erf(U1)</td>
<td>Gauss error function of U1</td>
</tr>
<tr>
<td>if(C,E,F) or</td>
<td>If C is true then E else F</td>
</tr>
<tr>
<td>if(C then E else F)</td>
<td></td>
</tr>
<tr>
<td>bess(U1,U2)</td>
<td>Bessel function of U1 of order U2</td>
</tr>
<tr>
<td>beta(U1,U2)</td>
<td>Beta function of U1 and U2</td>
</tr>
<tr>
<td>gamma(U1)</td>
<td>Gamma function of U1</td>
</tr>
<tr>
<td>normprob(U1)</td>
<td>Returns a value representing the cumulative probability level for the specified number of standard deviations (U1) away from the mean for a standard normal distribution. A negative argument represents values to the left of (less than) the mean.</td>
</tr>
<tr>
<td>normstds(P)</td>
<td>Returns a value representing the number of standard deviations away from the mean for the specified cumulative probability level (P) for a standard normal distribution. A negative output represents a value to the left of (less than) the mean.</td>
</tr>
<tr>
<td>tdist(P,nu)</td>
<td>Student’s t distribution for the specified cumulative probability level (P), with nu degrees of freedom.</td>
</tr>
<tr>
<td>tprob(X,nu)</td>
<td>Student’s t distribution, returns the cumulative probability for value X, with nu degrees of freedom.</td>
</tr>
<tr>
<td>ImpProb(Old,Target,Width)</td>
<td>Used for customized importance sampling for a given target probability and probability width. Takes a probability (Old) and returns a biased probability.</td>
</tr>
<tr>
<td>ImpWeight(Prob,Target,Width)</td>
<td>Used for customized importance sampling for a given target probability and probability width. Takes a biased probability (produced by ImpProb) and returns a weight.</td>
</tr>
<tr>
<td>ImpOld(Prob,Target,Width)</td>
<td>Used for customized importance sampling for a given target probability and probability width. Takes a biased probability (produced by ImpProb) and returns the unbiased probability. This is the inverse of ImpProb.</td>
</tr>
<tr>
<td>PDF_Value(D, P1)</td>
<td>Returns the value of the distribution at the specified cumulative probability level P1. The output has the same dimensions as the distribution being referenced.</td>
</tr>
<tr>
<td>PDF_CumProb(D, V)</td>
<td>Returns the cumulative probability of the distribution at the specified value V. The output is dimensionless.</td>
</tr>
</tbody>
</table>
### Function Description

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDF_Mean(D)</td>
<td>Returns the mean of the distribution. The output has the same dimensions as the distribution being referenced.</td>
</tr>
<tr>
<td>PDF_SD(D)</td>
<td>Returns the standard deviation of the distribution. The output has the same dimensions as the distribution being referenced.</td>
</tr>
<tr>
<td>PDF_CTE(D, P1)</td>
<td>Returns the conditional tail expectation of cumulative probability P1 (i.e., the expected value of the output given that it lies above a specified cumulative probability P1). This result represents the mean of the worst 100(1 – P1)% of outcome. The output has the same dimensions as the distribution being referenced.</td>
</tr>
<tr>
<td>PDF_DiscreteProb(D, V)</td>
<td>Returns the discrete probability of the distribution at the specified value V. The output is dimensionless. This function returns 0 for continuous distributions (i.e., it should only be applied to discrete distributions).</td>
</tr>
<tr>
<td>changed(Z)</td>
<td>Returns a condition. True if the argument has changed since the last update (e.g., timestep). False if the argument has not changed. See Note below.</td>
</tr>
<tr>
<td>occurs(T)</td>
<td>Returns a condition. True if the argument has occurred this update (e.g., timestep). False if the argument has not occurred.</td>
</tr>
</tbody>
</table>

**U1,U2:** Must be a unitless value. Can be a scalar or an array.

**D:** A distribution output.

**V:** A scalar value.

**X, Y:** A value. Can be a scalar or an array.

**E, F:** A value or a condition. Can be a scalar or an array.

**C:** Must be a condition. C can be an array or scalar; if it is an array, then the "if" test is done on a term-by-term basis and either E or F (or both) must be an array with the same set of array labels as C. If C is scalar, E and F can be either scalar, arrays of the same order (entire array is then selected), or one can be an array and one can be a scalar (in which case the scalar is treated as an array of identical values).

**P:** A value between 0 and 1, inclusive. Can be a scalar or an array.

**P1, Old, Target:** A scalar value between 0 and 1, inclusive.

**Width:** A scalar value between 10^-6 and 0.5, inclusive.

**nu:** A positive integer. Can be a scalar or an array.

**Z:** A value or a condition. Must be a scalar.

**T:** Must be a discrete event signal.

**Read more:** [Manipulating Arrays in Expressions Using Mathematical Operators](#) (page 799); [Specialized Functions That Operate on Distributions](#) (page 182).

**Note:** If the argument to the changed() function is a Run Property that conceptually cycles (e.g., Hour), and your timestep is such that the actual value remains unchanged (e.g., Changed(Hour), with a 1 day timestep), GoldSim will still consider that the argument has changed (since the fact that it remains unchanged is simply an artifact of timestepping).

**Read more:** [Understanding and Referencing Run Properties](#) (page 486); [Determining if an Event has Occurred](#) (page 417).
Note: The three importance sampling functions (ImpProb, ImpWeight and ImpOld) are used to carry out customized importance sampling of Stochastic elements.

Read more: Customized Importance Sampling Using User-Defined Realization Weights (page 1013).

Financial Functions

GoldSim provides a number of built-in financial functions.

The functions below are simple compounding functions that account for the time value of money and convert between present value, future value and annuities. These are dimensionless functions that act as multiplying factors on a currency amount (e.g., $) or currency rate (e.g., $/yr).

In these functions the first argument (int. rate) is the fractional interest per period (expressed as either a fraction or a percentage), and the second argument (#periods) is the number of periods:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ftop(int. rate, #periods)</td>
<td>Factor that when multiplied by a future amount returns the present value of that future amount.</td>
<td>$(1 + i)^n$</td>
</tr>
<tr>
<td>ptof(int. rate, #periods)</td>
<td>Factor that when multiplied by a present amount returns the future value of that present amount.</td>
<td>$(1 + i)^n$</td>
</tr>
<tr>
<td>atop(int. rate, #periods)</td>
<td>Factor that when multiplied by an annuity amount returns the present value of that annuity.</td>
<td>$\frac{(1+i)^n -1}{i(1+i)^n}$</td>
</tr>
<tr>
<td>atof(int. rate, #periods)</td>
<td>Factor that when multiplied by an annuity amount returns the future value of that annuity.</td>
<td>$\frac{(1+i)^n -1}{i}$</td>
</tr>
<tr>
<td>ptoa(int. rate, #periods)</td>
<td>Factor that when multiplied by a present amount returns the annuity of that present amount.</td>
<td>$\frac{i(1+i)^n}{(1+i)^n -1}$</td>
</tr>
</tbody>
</table>

int. rate (i): Must be greater than zero. Can be a scalar or an array.

#periods (n): Must be a positive scalar.

These functions assume discrete compounding. That is, the functions assume compounding once per period.

Since #periods represents the number of periods in these functions, it must be a unitless value. For example, in order to compute the annuity value (the annual payment amount) for a 15 year loan of $100,000 with an interest rate of 7% per year (compounded annually), you would write the following in an expression:

100 $ * ftop(0.07, 5)

GoldSim also provides more advanced financial functions for use in conjunction with the GoldSim Financial Module. These are described in the GoldSim Financial Module User’s Guide.
GoldSim provides a number of constants that can be used in equations. The built-in constants are:

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
<th>Units</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>amu</td>
<td>Atomic mass unit</td>
<td>kg</td>
<td>1.6605402e-027</td>
</tr>
<tr>
<td>c</td>
<td>Speed of light</td>
<td>m/s</td>
<td>299792458</td>
</tr>
<tr>
<td>e</td>
<td>Base of natural logarithms</td>
<td></td>
<td>2.71828182845905</td>
</tr>
<tr>
<td>ec</td>
<td>Elementary charge (electron)</td>
<td>s-amp</td>
<td>1.60217733e-019</td>
</tr>
<tr>
<td>Eps0</td>
<td>Permittivity of vacuum</td>
<td>s4-amp2/kg-m3</td>
<td>8.85418781762039e-12</td>
</tr>
<tr>
<td>ev</td>
<td>Electron-volt</td>
<td>kg-m2/s2</td>
<td>1.60217733e-019</td>
</tr>
<tr>
<td>G</td>
<td>Gravitational constant</td>
<td>m3/kg-s2</td>
<td>6.67259e-011</td>
</tr>
<tr>
<td>gee</td>
<td>Acceleration of gravity</td>
<td>m/s2</td>
<td>9.80665</td>
</tr>
<tr>
<td>h</td>
<td>Planck’s constant</td>
<td>kg-m2/s</td>
<td>6.6260755e-34</td>
</tr>
<tr>
<td>HgDens</td>
<td>Density of mercury</td>
<td>kg/m3</td>
<td>13595.08</td>
</tr>
<tr>
<td>k</td>
<td>Boltzmann constant</td>
<td>kg-m2/s2-K</td>
<td>1.380658e-23</td>
</tr>
<tr>
<td>me</td>
<td>Electron mass</td>
<td>kg</td>
<td>9.1093897e-31</td>
</tr>
<tr>
<td>mn</td>
<td>Neutron mass</td>
<td>kg</td>
<td>1.6749286e-27</td>
</tr>
<tr>
<td>mp</td>
<td>Proton mass</td>
<td>kg</td>
<td>1.6726231e-27</td>
</tr>
<tr>
<td>Mu0</td>
<td>Permeability of vacuum</td>
<td>kg-m/s2-amp2</td>
<td>1.25663706143592e-06</td>
</tr>
<tr>
<td>N</td>
<td>Avogadro’s constant</td>
<td>l/mol</td>
<td>6.0221367e+23</td>
</tr>
<tr>
<td>pi</td>
<td>Pi</td>
<td></td>
<td>3.14159265358979</td>
</tr>
<tr>
<td>R</td>
<td>Gas constant</td>
<td>kg-m2/s2-K-mol</td>
<td>8.31451</td>
</tr>
<tr>
<td>sigma</td>
<td>Stefan-Boltzmann constant</td>
<td>kg/s3-K4</td>
<td>5.67051e-008</td>
</tr>
<tr>
<td>Stemp</td>
<td>Standard temperature</td>
<td>K</td>
<td>273.15</td>
</tr>
<tr>
<td>Vmol</td>
<td>Molar gas volume at STP</td>
<td>m3</td>
<td>0.0224140972760918</td>
</tr>
<tr>
<td>WatDens</td>
<td>Density of water at STP</td>
<td>kg/m3</td>
<td>999.95</td>
</tr>
<tr>
<td>WatWt</td>
<td>Unit weight of water at STP</td>
<td>kg/m2/s2</td>
<td>9806.16</td>
</tr>
</tbody>
</table>

**Warning:** If there is an element name which conflicts with (i.e., is identical to) a constant name, the element name always has precedence. Hence, if you type an expression such as 3*R, and an element named R exists in the same scope, GoldSim will try to create a link to the element rather than using the built-in constant R, even if the element's dimensions are incompatible (such that the expression becomes invalid).
Creating Conditional Expressions

GoldSim provides a number of operators and functions which allow you to create conditional expressions. As used here, a conditional expression is an expression (or part of an expression) which evaluates to (produces) a condition (rather than a value). A condition has two states: True or False.

The following expressions are all conditional expressions:

\[ A > B \]
\[ A = B \]
\[ A > 5 \text{ AND } A < 10 \]
\[ !A \]

The operators that can be used in conditional expressions, along with their precedence, are summarized below:

<table>
<thead>
<tr>
<th>Precedence Level</th>
<th>Type</th>
<th>Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highest</td>
<td>parentheses</td>
<td>()</td>
</tr>
<tr>
<td></td>
<td>logical NOT</td>
<td>!, NOT</td>
</tr>
<tr>
<td></td>
<td>relational</td>
<td>&gt;, &lt;, &gt;=, &lt;=</td>
</tr>
<tr>
<td></td>
<td>equality, inequality</td>
<td>= (or ==), &lt;&gt;</td>
</tr>
<tr>
<td></td>
<td>logical AND</td>
<td>&amp;&amp;, AND</td>
</tr>
<tr>
<td>Lowest</td>
<td>logical OR</td>
<td></td>
</tr>
</tbody>
</table>

Conditional expressions are very useful, because they allow you to add if…then logic to your model. This can be accomplished by using GoldSim’s if function in an expression. For example, the expression

\[ \text{if}(\text{ETime} > 10 \text{ yr and ETime} < 20 \text{ yr then 33 else 44}) \]

would be interpreted as "If the elapsed time is between 10 years and 20 years, the value is equal to 33, otherwise it is equal to 44”.

Read more: Built-in Functions (page 124).

Complex if…then logic can be defined by nesting this function, or by using a Selector element. Conditional expressions are also of great use when modeling discrete events and consequences.

Read more: Selector Elements (page 278); Specifying Triggering Events (page 356).

Warning: The equality operator (= or ==) should be used with caution. In particular, you should generally not use this operator when comparing calculated values in your model (which could be affected by small round-off errors). Note that both ETime and Time are calculated values (since they are always represented internally in seconds) and hence they are also potentially susceptible to small roundoff errors if used in conjunction with the equality operator.

Note: Tool-tips for conditional outputs by default display true or false. You can specify what is displayed (e.g., on/off, yes/no, etc.) by editing the “Show condition result outputs as” field in the Results tab of the Options dialog (accessed from the main menu via Model|Options…).
Referencing Dates in Expressions

GoldSim allows you to reference dates in expressions by enclosing the date in quotation marks within an expression. The format for referencing a date is determined by the Regional and Language time/date settings specified by the operating system. For example, for a computer in the US, the following formats would all be valid ways for referencing October 1, 2009:

“10/1/2009”
“10-1-2009”
“October 1, 2009”
“1 October 2009”

However, if you entered the same expressions on an Australian or European computer, although the last two examples would be interpreted as the 1st day in October, the first two examples (“10/1/2009” and “10-1-2009”) would be interpreted as the 10th day in January. Similarly, on a German computer, “1 October 2009” would not even be recognized (GoldSim would display an error), but “1 Oktober 2009” would be correctly recognized and interpreted.

A date is internally converted and stored as a Julian time (in particular, the time since December 30, 1899 00:00:00). Hence, a date has dimensions of time.

Because dates are stored in this way, an expression could appear differently depending on the regional settings on the computer on which GoldSim is running. For example, if you entered a date (say October 1, 2009) into GoldSim on a computer with US settings like this:

and then saved the file and subsequently opened the file on a computer with Australian or European settings, the same expression would appear like this:

In both cases, GoldSim would interpret the date as October 1, 2010.

You would typically reference a date in conjunction with DateTime, as shown below:

\[
\text{DateTime is a Run Property, and represents the simulated calendar time.}
\]

Read more: Understanding and Referencing Run Properties (page 486).

Warning: You should take extra care when exchanging date information with other programs, such as Microsoft Excel, since they may treat Julian date references differently than GoldSim. Microsoft Excel, for example, actually uses a Julian reference date of December 31, 1899 00:00:00, but mistakenly adds an extra day into its calendar that did not actually exist (February 29, 1900). Fortunately, as a result of this, the effective Julian date reference for all times after March 1, 1900 in Excel is actually the same as that used by GoldSim, December 30, 1899 00:00:00.

Read more: Exchanging Date Information with a Spreadsheet (page 930).
Warning: If you enter a date using only two digits for the year (e.g., “10/1/09”), GoldSim interprets years less than or equal to 29 as being in the 21st century, and years greater than 29 as being in the 20th century. To avoid confusion, it is strongly recommended that you always use four digits to specify the year.

Read more: Setting the Basic Time Options (page 453).

Using Containers

Containers are perhaps the most important element type in GoldSim, because they allow you to create hierarchical, "top-down" models. This section provides the basic information required to use Containers in your models.

The example model BasicContainer.gsm in the General Examples/Containers folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) presents the basic concepts associated with the use of Containers.

Read more: Understanding Containers (page 95).

Like any element in GoldSim, you can edit the properties of a Container by double-clicking on it either in the graphics pane or the browser, or right-clicking anywhere inside the graphics pane when inside the Container, and selecting Properties…. The properties dialog looks like this:

By default, a Container has three tabs: General, Graphics, and Information.

Note: If you turn on some of the other features for a Container, additional tabs are added.

Within the General tab (shown by default), the top part of the dialog allows you to enter and edit the ID, Description and Appearance of the element.
Within the middle portion of the **General** tab, you can select which **Features** you would like the Container to have. By default, it has none of these features (it is a simple Container). If you click on the name of a feature, a brief description of the feature will appear in the **Details** window. Features are added and removed using the checkboxes to the left of the feature name.

**Read more:** Container Options and Features (page 133).

At the bottom of the **General** tab are options to **Save/Enable Results**. By default, a Container does not have any outputs of its own, and the first two checkboxes are grayed out. Under some conditions (e.g., making the Container conditional), it does have outputs, and in this case, these checkboxes become available for editing. The third option (to **Enable Time History Result Elements**) will be grayed out unless the Container includes some Time History Result elements (in which case it will be checked on). This allows you to globally disable all Time History Result elements inside the Container.

**Read more:** Controlling Result Flags for Elements in the Container (page 141).

The **Graphics** tab of the Container provides access to options for controlling the appearance of elements inside the Container (e.g., size of the graphics pane, and the manner in which the influences between elements are drawn).

**Read more:** Controlling the Appearance of the Graphics Pane in a Container (page 139).

The **Information** tab of the Container provides summary information for the Container, such as the total number of elements it contains, and the size of the results being saved inside the Container. It also provides the ability to recursively turn off result saving for all elements inside the Container.

**Read more:** Summary Information for a Container (page 140); Controlling Result Flags for Elements in the Container (page 141).

Within the middle portion of the **General** tab of the Containers dialog, you can select which **Features** you would like the Container to have:

![Features and Details windows](image)

By default, it has none of these features (it is a simple Container).

If you click on the name of a feature, a brief description of the feature will appear in the **Details** window. Features are added and removed using the checkboxes to the left of the feature name. Each of the available features is discussed in more detail in the sections below.

The default appearance of the Container changes depending on its features:
### Local and Global Containers

Each Container in a model is either global or local. The contents of a global Container can be "seen" (referenced) by any other element in the model. The contents of a local Container can typically only be referenced by other elements inside the Container.

By default, Containers are global. Containers can be localized by selecting the **Localization** feature in the Container dialog, or by right-clicking on the Container and selecting **Localize**. In addition, when you paste a Container, GoldSim will localize it if the names of elements inside the pasted Container conflict with element names at the location where it is being pasted.

You can recognize a localized Container in three ways:

- The triangle in the upper left hand corner of the Container's symbol in the graphics pane is red (instead of blue).
- The icon for the Container in the browsers and the default symbol for the Container in the graphics pane is a closed box (rather than an open box):

  ![Closed Box Icon](image)

  *This symbol is only used for simple Containers. Localized SubSystems use a different symbol but retain the red triangle.*
Using Containers

Chapter 4: Using the GoldSim Elements

• The tool-tip for the Container shows that it is Localized.

Read more: Localizing Containers (page 944).

One of the advanced options for a Container is to treat it as a SubSystem. If you select the Conditionality, Internal Clock, Looping Capability or Provide Resources feature for a Container, it will automatically be treated as a SubSystem (these features require this to be the case):

SubSystems are specialized Containers that are completely “self-contained” so that from the outside, they behave like a single element. What is meant by “self-contained” requires a somewhat detailed understanding of the order in which GoldSim carries out its calculations, and such an understanding is not required for the majority of users.


For most users, it is sufficient to note that being “self-contained” enables a SubSystem to take on some useful features and properties (e.g., conditionality, having an internal clock, and being able to loop). However, it also imposes some limitations with regard to how the SubSystem can be incorporated into certain types of systems (i.e., feedback loops).

Read more: Limitations on the Use of SubSystems in Feedback Loops (page 348).

For most users, there will be no need to treat a Container as a SubSystem.

Experienced users who would like to utilize advanced features such as Conditionality, Internal Clocks and Looping, or to more directly control the order in which GoldSim carries out its calculations, should make sure they clearly understand the limitations that SubSystems impose.

Conditionality is an advanced feature of GoldSim which allows parts of your model to dynamically activate and deactivate. By default, Containers are not conditional.

Containers can be made conditional by selecting the Conditionality feature in the Container dialog.

Conditional Containers
**Warning:** When you make a Container conditional, the **Treat as SubSystem** feature is also automatically selected (and cannot be deselected unless you first turn off **Conditionality**). That is, a Conditional Container, by definition, is treated as a SubSystem. Because a Conditional Container is treated as a SubSystem, this puts certain limitations on how Conditional Containers can be used.

*Read more: [Treating a Container as a SubSystem](page 135).*

When you make a Container conditional, a **Conditionality** tab is added to the Container dialog:

![Containter Conditionality Tab](image)

In addition, when a Container is made conditional, a number of inputs and outputs are added to the Container element itself. As a result, the Save Results checkboxes for the Container are no longer grayed out in the General tab (since a conditional Container has its own specific outputs).

*Read more: [Using Conditional Containers](page 899).*

In some situations, you may wish to use different timesteps in different portions of your model. This is useful, for example, if the processes represented in one portion of your model occur rapidly and hence require a small timestep to model accurately, while processes represented in the rest of your model occur much more slowly, such that a larger timestep can be used.

GoldSim allows you to assign a timestep that is smaller than the “global” timestep to one or more Containers. For example, your model could have a 10 day “global” timestep, but you could assign a 1 day timestep to a specific Container in your model that represents a system that changes rapidly and hence requires a smaller timestep to simulate accurately.

You can specify smaller timesteps for a Container by assigning the Container an “internal clock”. Containers can be assigned internal clocks by selecting the **Internal Clock** feature in the Container dialog.
Note: When you specify a Container as having an **Internal Clock**, you cannot also define it as having **Looping Capability** (these two options are mutually exclusive).

Warning: When you assign an internal clock to a Container, the **Treat as SubSystem** feature is also automatically selected (and cannot be deselected unless you first turn off **Internal Clock**). That is, a Container with an internal clock, by definition, is treated as a SubSystem. Because a Container with an internal clock is treated as a SubSystem, this puts certain limitations on how these Containers can be used.

Read more: **Treating a Container as a SubSystem** (page 135).

When you assign an internal clock to a Container, an **Internal Clock** tab is added to the Container dialog, and the field in this tab is used to define the manner in which GoldSim timesteps within the Container:

### Looping Containers

In some models, you may want to carry out an iterative calculation at each timestep. This might be useful, for example, if you have a coupled system of nonlinear equations that must be solved every timestep by iterating.

You can define a Container as a looping Container by selecting the **Looping Capability** feature in the Container dialog.

Note: When you specify a Container as having **Looping Capability**, you cannot also define an **Internal Clock** for the Container (these two options are mutually exclusive).
**Warning:** When you specify a Container as a looping Container, the Treat as SubSystem feature is also automatically selected (and cannot be deselected unless you first turn off Looping Capability). That is, a looping Container, by definition, is treated as a Subsystem. Because a looping Container is treated as a SubSystem, this puts certain limitations on how these Containers can be used.

*Read more:* [Treating a Container as a SubSystem](page 135).

When you specify a Container as a looping Container, a Looping tab is added to the Container dialog, and the fields in this tab are used to control how the Container carries out its loops:

![Looping tab in Container dialog](image)

*Read more:* [Using Looping Containers](page 963).

Providing Resources within a Container is an advanced feature of GoldSim. A Resource is something that has a limited supply (e.g., spare parts, fuel, skilled personnel, money) and is required in order for specified elements within the Container to carry out certain actions.

*Read more:* [Using Resources](page 839).

Resources can be provided to a Container by selecting the Provide Resources feature in the Container dialog.

**Warning:** When you provide Resources for a Container, the Treat as SubSystem feature is also automatically selected (and cannot be deselected unless you first turn off Provide Resources). That is, a Container with Resources, by definition, is treated as a SubSystem. Because a Container with Resources is treated as a SubSystem, this puts certain limitations on how the Containers behave.

*Read more:* [Treating a Container as a SubSystem](page 135).
When you provide Resources to a Container, a **Resources** tab is added to the Container dialog:

In some cases, you may want to protect the contents of a Container from being modified. GoldSim provides two options (sealing and locking) for doing so. Sealing a Container causes a warning message to be displayed whenever you attempt to modify the contents of the Container. Locking a Container completely prevents you from editing or modifying the contents of the Container. A password is required to unlock a Container.

You can protect the contents of a Container by selecting the **Protection** feature in the Container dialog. When you do so, the following dialog for specifying how you would like to protect the Container is displayed:

**Protected Containers**

The **Graphics** tab of the Container dialog provides access to options for controlling the appearance of elements inside the Container:

**Read more:** Sealing and Locking Containers (page 143).

The **Graphics** tab of the Container dialog provides access to options for controlling the appearance of elements inside the Container:
The top portion of the dialog (the Graphics Pane section) is used to define the size of the graphics pane, a background color for the graphics pane, and whether or not a grid is displayed.

**Read more:** Adjusting the Size of the Graphics Pane (page 425); The Graphics Pane Grid and Background (page 423).

The bottom portion of the dialog (the Influences section) is used to specify the default shapes for influences in the Container, and to define whether and in what manner influences within the Container are filtered (hidden). The options in this portion of the dialog are quite important, as they can be used to ensure that your models are easier to view and understand.

**Read more:** Links and Influences (page 80); Editing the Appearance of Influences (page 426); Filtering Influences (page 430).

**Note:** The graphical properties are of particular interest for the Model Container, as they allow you to control the appearance of the entire model. You can view the properties of the Model Container by double-clicking on it in the browser or by right-clicking anywhere in the graphics pane (when viewing the Model Container) and selecting **Properties**.

### Summary Information for a Container

The **Information** tab of the Container dialog provides access to summary information regarding the Container:
The dialog provides some useful summary statistics for the Container, including the number of model elements and graphical components and the degree of subcontainment. The number of levels of subcontainment does not refer to the number of Containers; it refers to the number of hierarchical levels of Containers. For example, if A contained Containers B and C, A would have 1 level of subcontainment; if A contained B, and B contained C, A would have 2 levels of subcontainment.

**Note:** These properties are of particular interest for the top level container (the Model Container), as they tell you the size of the entire model. You can view the properties of the Model Container by double-clicking on it in the browser.

The dialog also indicates the total size of the results being saved for all elements within the Container. This information can be very useful in helping you to manage the size of the model file.

The bottom of the dialog contains some advanced options to allow you to deactivate the results flags for all elements contained within the Container.

This is described in detail in the next section.

Before you run a model, you select the results that you wish to save. You can save **Final Values** (the values at the end of each realization in the simulation) and/or **Time History** (the value at selected timesteps throughout the simulation). Typically, these are specified by separately setting flags for each element (and each output).

In some cases, however, you may want to change these settings for all the elements in a Container at once (i.e., globally deactivate the result flags for all elements inside the Container).

You can do this using the two buttons located at the bottom of the **Information** tab or the Container dialog:
Note: The Time Histories button is only available if you are running a single realization. If you are running multiple realizations, the state of each element’s Time History flags are determined by whether or not the element is referenced by an active Time History Result element. If it is, the flag is turned on. If it is not, the flag is turned off. In both cases, the flag is grayed out and cannot be manually changed.

Read more: Specifying Results to be Saved (page 494).

To globally turn off saving of all of the results of a particular type in a Container, simply press the button. A dialog will be displayed asking you to confirm that you wish to turn off the results.

Of course, once you turn off results in a Container, you can subsequently enter the Container and set some of the elements’ results flags back on.

Note: This capability is of particular interest for the Model Container, as it can be used to globally turn off all result options for the entire model.

By default, you cannot turn off results for outputs which are linked to Result elements (unless the Result element is a Time History Result element that has been disabled). That is, any output linked to a Result element is automatically saved, even if you choose to turn off results using the buttons described above. This behavior can be changed in Time History Result elements (so the outputs linked to the Result element are not automatically saved) by disabling the element.

Read more: Disabling a Time History Result Element (page 635).

You can choose to globally disable all Time History Result elements in a Container by clearing the Enable Time History Result Elements checkbox at the bottom of the General tab of the Container dialog.
Sealing and Locking Containers

GoldSim provides two options to protect the contents of a Container from being modified: sealing a Container and locking a Container.

Sealing a Container is the weakest of the two methods. Sealing causes a warning message to be displayed whenever you attempt to modify the contents of the Container, but does not prevent you from modifying the contents if you choose to “break” the seal. Locking a Container completely prevents you from editing or modifying the contents of the Container without providing a password.

Sealing is the lowest level of protection you can place on a Container. Sealing a Container causes a warning message to be displayed whenever you attempt to modify the contents of the Container, but does not prevent you from modifying the contents if you choose to “break” the seal. It is typically used simply to remind users that a particular Container has been verified and/or checked in some manner and hence should not be edited. It also can be used to verify that the contents of a particular Container has not been modified.

You can seal the contents of a Container by selecting the Protection feature in the Container dialog. When you do so, the following dialog for specifying how you would like to protect the Container is displayed:

![Sealing & Locking dialog]

Your Windows user name will be inserted by default into the Name field (which you can subsequently edit).

By default, the option to **Seal Container** will be selected. The Container can then be sealed by pressing the **OK** button. Upon doing so, the following dialog will be displayed:

![GoldSim Dialog]

When a Container is sealed, the Container and all of its contents appear grayed out when viewed in a browser. In addition, the tool-tip for the Container indicates that it is sealed, and the Details section for the Container dialog displays the name of the person who sealed the Container, along with the date and time that the Container was sealed.

Once a Container has been sealed, you can make “cosmetic” changes to the contents (moving elements around in the graphics pane, adding text, graphics or images to the Container), but if you try to make any other kind of change (e.g.,
changing the inputs to an element, adding an element) GoldSim will provide a message warning you (twice!) that the seal will be “broken” if you continue.

Once a seal is broken, this new status (along with who broke the seal and when it was broken) is also displayed in the Details section of the dialog for the Container.

Once a seal is broken, you can choose to seal the Container again (by pressing the Protection checkbox again).

You can remove (as opposed to break) the seal from a Container by pressing the Protection checkbox.

**Warning**: If a seal is removed (rather than broken), no record of this is written to the Details section of the dialog for the Container.

**Note**: If you seal a Container which contains other Containers, all of the Containers within the parent Container are also sealed. If you then remove the seal from the parent Container, however, the other Containers within it remain sealed.

**Warning**: The values of Data elements that are located within a sealed Container and are controlled via a Dashboard control can be changed via the control without breaking the seal. Dashboard controls are discussed in the GoldSim Dashboard Authoring Module User’s Guide.

If you need to protect the contents of a Container such that they cannot be edited at all, you should lock the Container.

**Locking a Container**

Locking is the highest level of protection you can place on a Container. Locking a Container completely prevents the user from editing or modifying the contents of the Container without providing a password.

You can lock the contents of a Container by selecting the Protection feature in the Container dialog. When you do so, the following dialog for specifying how you would like to protect the Container is displayed:

Your Windows user name will be inserted by default into the Name field (which you can subsequently edit).
By default, the option to Seal Container will be selected. To lock the Container, select the Lock Container radio button. When you do so, the Password and Verify fields will become editable. To lock the Container, enter a password, verify it, and press the OK button. Upon doing so, the following dialog will be displayed:

When a Container is locked, the Container and all of its contents appear grayed out when viewed in a browser. In addition, the tool-tip for the Container indicates that it is locked, and the Details section for the Container dialog displays the name of the person who locked the Container, along with the date and time that the Container was locked. Moreover, whenever you are inside a locked Container, the cursor is changed to the “Lock Cursor”, to remind you that the Container is locked.

Note also that checkboxes on the property dialog are grayed out (except for the Protection checkbox). In fact, the property dialogs of all elements within a locked Container will also be uneditable. In addition, when viewing the contents of a locked Container, most menus are disabled. This is because you cannot make any changes at all to the contents of a locked Container (or to the properties of the locked Container itself). You can view the contents of a locked Container (and all the properties of the elements within the Container), but the contents of the Container cannot be edited until the Container is unlocked.

To unlock a locked Container, click on the Protection checkbox on the properties dialog for the Container. You will be prompted for a password.

Enter the password and press OK. The Container and its contents will then be available for editing.

Note: When you lock a Container, you can choose not to specify a password (leave the password blank). If you do this, the Container will unlock as soon as you press the Protection checkbox (without prompting you for a password).

Note: The Properties page for Result elements in locked Containers cannot be edited. However, when displaying a result via a Result element that is inside a locked Container, you can edit the style (e.g., headers, axes titles) via the Edit Style button on the display page.
Warning: The values of Data elements that are located within a locked Container and are controlled via a Dashboard control can be changed via the control with unlocking the Container. Also, the values of Data elements that are located within a locked Container and are specified as being Scenario Data can be changed via the Scenario Manager without unlocking the Container. Dashboard controls are discussed in the GoldSim Dashboard Authoring Module User’s Guide.

Read more: Creating and Using Result Elements (page 570); Scenario Data: Defining Inputs for Different Scenarios (page 507).

Overview of GoldSim Element Types

GoldSim provides a wide variety of elements from which you can build your models. These elements can be divided into a number of categories. When you choose to insert an element into GoldSim, the Insert menu sorts the elements into these categories.

The various categories are discussed in general terms below. Most of the elements are then discussed in detail in the remainder of this chapter. Several of the more advanced elements are discussed in subsequent chapters.

Input Elements

Data elements allow you to specify a single scalar value (e.g., the discount rate) or an array of related values (e.g., the salaries of each individual in a group). Stochastic elements allow you to specify that a particular value is uncertain by defining it as a probability distribution. Time Series elements allow you to specify a time series of a value (e.g., monthly rainfall rates, quarterly cashflows). Lookup tables allow you to create response surfaces. History Generators provide a mechanism for creating stochastic time series based on statistical inputs.
Stock Elements

Stocks are elements that impart inertia and memory to a system, and hence are responsible for internally generating the dynamic behavior of many systems. In GoldSim, Stocks, a second element category called Delays, and a special function element (Convolution) are collectively referred to as dynamic elements.

Mathematically, Stocks are time integrals (i.e., they integrate flows or signals over time). As a result, they have the property that their outputs are determined by the historical (integrated) value of their inputs.

GoldSim provides three Stock elements: Integrators, Reservoirs and Pools.

Pools and Reservoirs are intended to accumulate flows of materials (e.g., cash, water, widgets). Integrators simply integrate signals (e.g., the height of a thrown ball could be computed by integrating the velocity of the ball).

Function Elements

Functions compute outputs based on the current (instantaneous) values of their inputs. Hence, unlike Stocks, Functions do not impart inertia or memory to a system. Rather, they instantaneously transform or convert inputs to outputs.

GoldSim provides a wide variety of Function elements.
As can be seen in the menu, Function elements are subdivided into three types: Standard Functions, which include Expressions and Selectors, Logical Functions for manipulating logical expressions, and External Application Functions for linking external applications such as spreadsheets into GoldSim.

**Read more:** Using Basic Function Elements (page 274); Script Elements (page 860); Solving Convolution Integrals (page 817); Using External Application Elements (page 910).

**Event Elements**

Event elements allow you to superimpose the occurrence and effects of discrete events onto continuously varying systems.

This allows for the realistic simulation of such things as financial transactions, accidents, system failures, storms, labor strikes and lawsuits.

There are ten elements that can be used to create and react to discrete events. Eight of these are available under the Events submenu. Two additional elements are available under the Delays submenu.
Delays simulate processes that delay continuous or discrete signals and flows. That is, their outputs lag their inputs. Therefore, like Stocks, their outputs are computed based on the historical values of their inputs.

As a result, Delays also impart inertia and memory to a system, and can internally generate dynamic behavior within a system.

There are four Delay elements in GoldSim: Material Delays delay continuous flows of material; Information Delays delay continuous flows of information; Event and Discrete Change Delays delay discrete signals.

Result elements provide a convenient way to collect, analyze and display results. While you can look at the results associated with any element by finding the element and right-clicking on it, Result elements provide a mechanism for making key results readily accessible in a location separate from the element itself.

GoldSim provides four types of Result elements, one for each of the basic types of results that can be displayed in GoldSim.
Overview of GoldSim Element Types

Chapter 4: Using the GoldSim Elements

Differentiating Between Material and Information Flow

Read more: Creating and Using Result Elements (page 570).

The purpose of any dynamic model is to describe the movement, evolution and/or transformation of materials and/or information. Material can be thought of as “stuff”: things that are tangible, like widgets, water, dirt, buildings or cash. Information is intangible, and represents things like temperature, interest rates, prices, or a person’s perceptions.

Many models simulate the movement or transformation of materials. A water resources model simulates the movement of water through a system. A financial model simulates the movement and transformation of assets (e.g., cash, property) within a system. Other models simulate the evolution of information (e.g., the status of a project, a person’s perception of or degree of belief in something, the price of a commodity).

Many models simulate both material and information flows. For example, a supply chain model simulates the movement and transformation of parts and products (materials) as well as the perceptions of purchasing managers and other individuals (information).

Although the movement/evolution/transformation of materials and information within a system are similar in many respects, they have an important difference. In particular, material is conserved as it moves through a system, while information generally is not. Treating material and information flows in the same manner within a model can lead to conceptual and mathematical errors. It can also add to the complexity (and lack of transparency) in a model.

As a result, GoldSim explicitly distinguishes between the movement/evolution/transformation of material and information by providing separate elements that are intended only to operate on a particular type of input. This ability facilitates the construction of internally-consistent and transparent models.

GoldSim provides this differentiation for two kinds of elements:

**Stocks:** GoldSim provides three basic Stock elements. Reservoirs and Pools are intended to integrate/accumulate material flows, and represent things such as reservoirs of water, account balances, and inventories. Integrators are intended to simply integrate information. For example, integrating a time series of the velocity of a projectile using an Integrator would yield the distance that the projectile had traveled.

**Delays:** GoldSim provides two Delay elements for delaying continuous signals. Material Delays are intended to accept flows of material (e.g., gal/day or $/yr), and conserve the material in transit, outputting a flow. An
Using Basic Input Elements

Information Delay element accepts any kind of signal (which is generally not a flow), and does not enforce conservation of the signal.

While building your models, you should think carefully about whether you are operating on material or information, and use the appropriate element. As a general rule, if it is tangible (i.e., can be quantified in terms of mass, volume, or number of items), it is a material.

**Note:** Energy, although it is not tangible, should also be treated as a material, since it is always conserved. An example is a model that simulates heat transport through a system. In this case, the transport of heat is analogous to the transport of mass, and hence heat (energy) should be treated as a material within GoldSim. The dimensions of energy are $ML^2T^{-2}$.

**Note:** Even though their value is not necessarily conserved (e.g., due to currency fluctuations or fluctuating market values), cash and other assets such as stocks should always be thought of as materials (since they represent quantities of bills or shares, which are conserved). That is, the quantity of shares that you own is a material; the value of those shares is information.

Using Basic Input Elements

This section describes the two basic input elements: Data elements and Stochastic elements. The three other types of input elements (Time Series, Lookup Table and History Generator) are described separately.

**Read more:** Using Time Series Elements (page 186); Using Lookup Table Elements (page 294); Generating Stochastic Time Histories (page 826).

**Data Elements**

Data elements are intended to represent the constant inputs in your model.

A Data element can represent both values and conditions (i.e., True/False), and can represent a single scalar datum, or an array of data.

The properties dialog for a Data element looks like this:
As can be seen, the dialog is very simple. Other than the Element ID, Description and Display Units, the Data element property dialog has only a single input field (labeled Data Definition). Because Data elements are meant to represent constants in your model, typically you will enter a number into this field.

If you specify that the output of the Data element is a vector or a matrix, the dialog will be slightly modified (in order to allow you to define the members of the array). Instead of displaying an input field, it displays a button for defining the array:

Read more: Using Vectors and Matrices (page 782).

Note: Although you can enter expressions (with links) into the Data Definition field for a Data element, it is generally recommended that you only enter numbers in this field (and use an Expression element when you need to define equations with links to other elements). Doing so helps to make your models more transparent, since the Data symbol will visually indicate to anyone viewing the model that the element represents a constant.

Data elements have a single output, which can be a value or a condition, and can be specified as a scalar, a vector or a matrix. You can specify these attributes by pressing the Type... button. By default, a new Data element is a scalar, dimensionless value.

Data elements also have a Data Source field that allows you to download data directly from an ODBC-compliant database.

Read more: Linking Elements to a Database (page 1031).

If you are using GoldSim’s advanced scenario feature, and have defined at least one scenario, Data elements include an additional field that allow you to define the element as Scenario Data:
You can save the results for a Data element by clicking Final Values and/or Time Histories.

As shown below, the browser view of a Data element shows a single input (referred to as the Definition), and a single output:

Note: Element inputs and outputs are only shown in the browser if you choose to Show Element Subitems (accessed via the browser context menu by right-clicking in the browser).

Stochastic Elements

Stochastic elements allow you to explicitly represent uncertainty in the input data for your model. GoldSim uses the Monte Carlo method to sample Stochastic elements in order to carry out probabilistic simulations.

The probabilistic simulation techniques used by GoldSim are discussed in general terms in Appendix A “Introduction to Probabilistic Simulation”. If you are unfamiliar with probabilistic modeling, you should read this appendix before using a Stochastic element.

The dialog for a Stochastic element looks like this:
The Edit... button is used to define the distribution type and specify its parameters. The two Mode radio buttons are used to control when a Stochastic element is sampled.

The More button expands the dialog to provide access to a number of advanced features:

The advanced options include correlating one Stochastic to another, directly specifying a percentile level to use (rather than random sampling), modifying the sampling algorithm (importance sampling), linking the element to a database, and specifying how the Stochastic is to be evaluated when running deterministic simulations.

Read more: Linking Elements to a Database (page 1031).

Note: The More button becomes a Less button when the dialog is expanded. If any of these options are selected, the Less button is removed (the dialog remains expanded) the next time you open it.

Stochastic elements have four outputs:

- the realized (sampled) value, which is the primary output (and hence has the same name as the element);
- the probability density of the realized value (Probability_Density);
- the cumulative probability of the realized value (Cumulative_Probability); and
• a complex output representing all the distribution information (Distribution).

The first three outputs of Stochastic elements are always scalar or vector values (unless the Stochastic is defined as a Boolean, in which case the primary output is a condition). By default, a new Stochastic element is dimensionless. You specify the dimensions (display units) of the primary output of a Stochastic element in the Display Units field. The dimensions of the Probability_Density output are the inverse of those of the primary output (e.g., if the primary output has units of meters, the Probability_Density has units of inverse meters). The Cumulative_Probability output is dimensionless.

The fourth output of a Stochastic element (Distribution) is a complex output that represents all the statistical information necessary to define a probability distribution. It can only be used in several specialized locations (e.g., SubModel Interface, inputs to specialized distribution functions, inputs for an Externally-defined distribution, specialized input to Timed Events and Event Delays). It can also be viewed directly in a Distribution Result display in order to display the analytical form of the distribution.

Read more: Creating the Input Interface to a SubModel (page 980); Specialized Functions that Operate on Distributions (page 182); Externally-Defined Distribution (page 167); Timed Event Elements (page 365); Modeling Event Delays with Dispersion (page 379); Adding a Distribution Output to a Distribution Result (page 663).

You can save the results for a Stochastic element by clicking Final Values and/or Time Histories in the properties dialog. Checking one of these causes all three outputs to be saved as results. If you wish to save only one or two of these as results, you can use the context menu of the output (accessed via a right-click in the browser) to turn on or off one or more of the three outputs. In this case, the checkbox in the properties dialog will become a box instead of a check (indicating that only some of the results will be saved). Note that in order to see outputs in the browser you must first ensure that Show element subitems is selected by right-clicking anywhere in the browser.

Stochastic elements are discussed in detail in the sections below.

The Edit ... button on the Stochastic dialog accesses a dialog for specifying the probability distribution defining the element.
Using Basic Input Elements

The drop-list in the upper left-hand corner of the dialog contains a list of all of the distributions provided by GoldSim.

Directly below this field, you enter the parameters defining the selected distribution (the parameters differ depending on the distribution type). These can be constants, links or expressions.

A preview of the distribution chart is displayed in the upper right-hand corner of the dialog. Note that after edited the distribution parameters, you must press Apply to update the chart.

The statistics for the distribution (Mean, Standard Deviation, Skewness, and Kurtosis) are shown directly below the distribution's parameters. The significance of these four statistics is described in Appendix A, "Introduction to Probabilistic Simulation”.

The Percentile button displays the percentiles of the distribution:

The Calculator section of the dialog allows you to compute the value associated with a particular percentile or the percentile associated with a particular value:
Enter the percentile in the Cum. Prob. field and the corresponding value will be displayed in Value OR Enter the value in the Value field and the corresponding percentile will be displayed in Cum. Prob.

The Calculator also displays the Probability Density and the **Conditional Tail Expectation** for the specified Cumulative Probability/Value pair. The Conditional Tail Expectation is the expected value of the output given that it lies above a specified Cumulative Probability. That is, it represents the mean of the worst 100(1 - α)% of outcomes, where α is the specified Cumulative Probability.

**Note:** Calculation of the Conditional Tail Expectation is discussed in detail in Appendix B.

The toolbar at the top of the dialog is used to control the distribution plot. The functions of the various buttons are as follows:

<table>
<thead>
<tr>
<th>Button</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="pdf.png" alt="PDF" /></td>
<td>Display the preview as a PDF (probability density function):</td>
</tr>
</tbody>
</table>

Note that if **Fill Area** is checked below the preview, the area below the PDF line is filled in (this applies to CDF and CCDF previews also).
Using Basic Input Elements

**Button**  
**Action**

CDF

Display the preview as a CDF (cumulative distribution function):

Note that for a CDF, if Show Marker is checked below the preview, the particular value/percentile entered in the Calculator section is indicated on the plot.

CCDF

Display the preview as a CCDF (complementary cumulative distribution function):

Display a full chart of the distribution. (You can also display the full chart by double-clicking on the preview or pressing Ctrl+Shft+T). (When viewing the full chart, pressing the button again returns to the Distribution dialog.) A full chart can be customized to a much greater degree than a preview chart (e.g., you can add headers and footers; axis labels, etc.).

Read more: Viewing a Distribution Chart (page 647).

Note: You can toggle through the three types of distribution displays (PDF, CDF, CCDF) using the PgUp and PgDn keys.

**Stochastic Distribution Types**

The types of probability distributions supplied by GoldSim are summarized below:

<table>
<thead>
<tr>
<th>Distribution Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beta</td>
</tr>
<tr>
<td>Generalized Beta</td>
</tr>
<tr>
<td>BetaPERT</td>
</tr>
<tr>
<td>Binomial</td>
</tr>
</tbody>
</table>
The fields that appear in the Parameters section of the dialog will change with the distribution type. As a general rule, you should not specify the parameters of a distribution as a function of time, since GoldSim does automatically resample the distribution unless you specifically instruct GoldSim to do so.

Read more: Controlling When a Stochastic Element is Sampled (page 177).

The required inputs for each of the distributions types are summarized below. An example file which uses all of the Stochastic elements (Distributions.gsm) is in the General Examples/Stochastic folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu). The mathematical details of each of the distributions are provided in Appendix B.

The Beta distribution is a continuous distribution used to represent the uncertainty in the probability of occurrence of an event given some data regarding observations of the event occurrence in previous experiments. As such, the Beta distribution represents the probability distribution of a frequency, and therefore always ranges between 0 and 1.

It is defined by specifying results of an experiment or a number of trials (e.g., a coin flip) in terms of Number of Successes and Number of Failures:

<table>
<thead>
<tr>
<th>Distribution Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
</tr>
<tr>
<td>Cumulative (and log-Cumulative)</td>
</tr>
<tr>
<td>Discrete</td>
</tr>
<tr>
<td>Exponential</td>
</tr>
<tr>
<td>Externally-defined</td>
</tr>
<tr>
<td>Extreme Probability</td>
</tr>
<tr>
<td>Extreme Value</td>
</tr>
<tr>
<td>Gamma</td>
</tr>
<tr>
<td>Log-Normal</td>
</tr>
<tr>
<td>Negative Binomial</td>
</tr>
<tr>
<td>Normal</td>
</tr>
<tr>
<td>Pareto</td>
</tr>
<tr>
<td>Pearson Type III</td>
</tr>
<tr>
<td>Poisson</td>
</tr>
<tr>
<td>Sampled Results</td>
</tr>
<tr>
<td>Student’s t</td>
</tr>
<tr>
<td>Triangular (and log-Triangular)</td>
</tr>
<tr>
<td>Uniform (and log-Uniform)</td>
</tr>
<tr>
<td>Weibull</td>
</tr>
</tbody>
</table>

Beta Distribution

The Beta distribution is a continuous distribution used to represent the uncertainty in the probability of occurrence of an event given some data regarding observations of the event occurrence in previous experiments. As such, the Beta distribution represents the probability distribution of a frequency, and therefore always ranges between 0 and 1.

It is defined by specifying results of an experiment or a number of trials (e.g., a coin flip) in terms of Number of Successes and Number of Failures:
Both inputs must be positive values. The displayed distribution represents the underlying probability of a Success. For example, in the screen shown above, if we define Success as heads on a coin flip, and Failure as tails, the distribution represents the uncertainty in the probability of obtaining heads on the next flip of the coin. If the number of trials is small, the uncertainty is high. As the number of trials increases, the uncertainty in the outcome decreases (and in the example of a coin flip, if the coin is unbiased, the probability of heads would collapse to 0.5).

**Note:** The Beta distribution must always be dimensionless, as it represents a frequency (otherwise GoldSim will display an error). However, GoldSim also provides a generalized form of the Beta (described below) that allows you to specify a distribution’s dimensions (e.g., time). Such a distribution is defined in a different way (using a mean, standard deviation, minimum and maximum). The Generalized Beta distribution therefore provides a flexible way to specify a probability distribution over a particular range.

**Generalized Beta Distribution**

The Generalized Beta distribution provides a flexible way to specify a probability distribution over a particular range (i.e., with a specified minimum and maximum). It is commonly used within project management simulations to represent the time to complete a task.

It is defined by specifying a **Mean, Standard Deviation, Minimum** and **Maximum**:
Note: The theoretical maximum standard deviation (SD) for a beta distribution (ranging from 0 to 1) is equal to \(\sqrt{\text{mean} \cdot (1 - \text{mean})}\). However, with such a high SD, the beta distribution would have a “spike” of probability density at both its upper and lower bounds. With a somewhat lower SD, there will only be a spike at the limit closest to the mean. GoldSim restricts the SD to 0.6 of the maximum theoretical value in order to avoid having any spikes in the density function. If you need probability distributions that combine discrete spikes with continuous curves, you should construct this explicitly (by combining multiple distributions using an Expression, or by using a Cumulative distribution).

**BetaPERT Distribution**

The BetaPERT distribution is a specialized version of the Beta distribution and is often used for project risk analysis. Mathematically, it is a Beta distribution. While a Generalized Beta Distribution requires four parameters, a BetaPERT requires only three. A particular formula (inherent to the BetaPERT) determines the mean of the distribution (and hence its shape).

By default, it is defined by specifying either a **Minimum (0th percentile)**, **Most Likely** and **Maximum (100th percentile)**:

---

Read more: [Cumulative Distribution](page 164).
Alternatively, if you check the 10% / 90% checkbox, you can specify the 10th percentile, Most Likely and 90th percentile.

**Read more:** Generalized Beta Distribution (page 160).

The Binomial distribution describes the probability of a certain number of instances, given a (dimensionless) batch size and a probability of occurrence. As such, by definition, a Binomial distribution is dimensionless.

For example, you could use a Binomial distribution to define a probability distribution for the number of defective widgets produced, given a particular batch size (e.g., 10) and the probability of any given widget being defective (e.g., 0.05).

It requires two inputs, the **Batch Size** (which should be an integer) and the **Probability**. Like the Poisson distribution, it is a discrete distribution containing only integer numbers.
Note: Since this is a discrete distribution, the PDF view does not actually display a probability density function. Instead, it displays a probability mass function.

**Boolean Distribution**

The Boolean distribution takes on one of two values: True or False. It requires a single input, **Probability of True**. It is useful for representing binary systems which have a given probability (e.g., flipping a coin):

Note that when plotted, GoldSim shows a True condition as 1, and a False condition as 0.
Note: Since this is a discrete distribution, the PDF view does not actually display a probability density function. Instead, it displays a probability mass function.

**Cumulative Distribution**

The Cumulative distribution is used to define custom continuous distributions. You define the distribution by specifying [cumulative probability, value] pairs. A given pair (defined by a **Prob. Level** and a **Value**) implies that the cumulative probability of the **Value** is equal to the specified **Prob. Level**:

In this example, the cumulative probability of the value being less than or equal to 4 is 90%.

You add and remove pairs using the **Add** and **Remove** buttons (new pairs are inserted below the selected pair). In addition, if you place the cursor in the grid, the following hotkeys can be used:

<table>
<thead>
<tr>
<th>Keys</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl+Enter</td>
<td>creates a new row below the cursor</td>
</tr>
<tr>
<td>Ctrl+Shift+Enter</td>
<td>creates a new row above the cursor</td>
</tr>
<tr>
<td>Ctrl+Backspace</td>
<td>deletes the current row</td>
</tr>
</tbody>
</table>

By definition, the first cumulative probability specified must be 0 and the last cumulative probability specified must be 1. The **Prob. Level** and the **Value** of the pairs cannot decrease as you move downward through the list. If the Prob. Levels decrease as you move down the table, GoldSim will sort the rows so that they are in increasing order when you close the dialog or press **Apply**. If the Values decrease as you move downward, GoldSim will warn you and ask you to fix the distribution.

Note: Specifying two or more Prob Levels with the same number results in a horizontal line in the CDF. Specifying two or more Values with the same number results in a vertical line in the CDF (and an infinite spike in the PDF).
You can also define the distribution as a log-cumulative (by checking the Log box). Whereas in a cumulative distribution, the density between values is constant (i.e., the distribution between values is uniform), in a log-cumulative, the density of the log of the value is constant (i.e., the distribution between values is log-uniform). If Log is checked, all Values for the distribution must be positive. Log-Cumulative distributions are often applied to quantities with large (order-of-magnitude) uncertainties.

**Note:** The values and cumulative probabilities must be entered as numbers and cannot be specified using links or expressions. The values are assumed to be entered in the Display Units for the element.

**Note:** You can copy and paste the entries for a Cumulative distribution from a spreadsheet. To do so, select the two columns in the spreadsheet (Probability Level and Value), and copy them to the clipboard. Then place the cursor in the upper left-hand corner of the grid (i.e., the first Probability Level) and paste (by pressing Ctrl+V).

### Discrete Distribution

A Discrete distribution is used to specify a discrete (as opposed to continuous) probability density function. In a Discrete distribution, probabilities are assigned to discrete values. The probability of a value between the specified values is zero.

You define the distribution by specifying [probability, value] pairs. A given pair (defined by a Probability and a Value) implies that the probability of the Value is equal to the specified Probability:

![Discrete Distribution](image)

*In this example, the probability of the value being equal to 3 is 50%.*

You add and remove pairs using the Add and Remove buttons (new pairs are inserted below the selected pair). In addition, if you place the cursor in the grid, the following hotkeys can be used:
Using Basic Input Elements

<table>
<thead>
<tr>
<th>Keys</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl+Enter</td>
<td>creates a new row below the cursor</td>
</tr>
<tr>
<td>Ctrl+Shift+Enter</td>
<td>creates a new row above the cursor</td>
</tr>
<tr>
<td>Ctrl+Backspace</td>
<td>deletes the current row</td>
</tr>
</tbody>
</table>

By definition, the probabilities must sum to one. The **Values** do not need to be in any particular order (GoldSim will automatically sort them in increasing order when the data is applied). However, if the Probabilities do not sum to one, GoldSim will warn you and ask you to fix the distribution.

**Note:** The values and probabilities must be entered as numbers and cannot be specified using links or expressions. The values are assumed to be entered in the Display Units for the element (which are displayed in the header for the editing grid).

**Note:** You can copy and paste the entries for a Discrete distribution from a spreadsheet. To do so, select the two columns in the spreadsheet (Probability and Value), and copy them to the clipboard. Then place the cursor in the upper left-hand corner of the grid (i.e., the first Probability) and paste (by pressing Ctrl+V).

**Note:** Since this is a discrete distribution, the PDF view does not actually display a probability density function. Instead, it displays a probability mass function.

**Exponential Distribution**

The Exponential distribution is a continuous distribution, typically used to model the time to complete a task or reach a milestone.

It is defined by a single parameter: the **Mean** of the distribution (which must be positive).
The Externally-defined distribution is a specialized distribution that is defined externally by a Distribution output:

The Externally Defined Distribution field only accepts a Distribution output. Distribution outputs are complex outputs that represent all the statistical information necessary to define a probability distribution. They can only be produced by another Stochastic element, a Spreadsheet element, or a SubModel.

The primary purpose of this distribution type is to connect a Stochastic to a distribution that has been imported via a Spreadsheet element.

Read more: Importing Stochastic Element Definitions from a Spreadsheet (page 918).
However, there are also several other specialized uses, such as exchanging distribution information with a SubModel, and to support dynamically revising distributions using simulated Bayesian updating.

**Read more:** [Creating the Input Interface to a SubModel](#) (page 980); [Creating the Output Interface to a SubModel](#) (page 984); [Dynamically Revising Distributions Using Simulated Bayesian Updating](#) (page 1016).

The Extreme Probability distribution is a continuous distribution used to represent the uncertainty in the highest or lowest probability level if any given distribution is sampled N times. As such, the Extreme Probability distribution represents the probability distribution of a probability level, and therefore always ranges between 0 and 1.

It is defined by specifying the **Number of samples**, and whether you are interested in the probability level for the **Maximum value** or the **Minimum value**:

![Extreme Probability Distribution](image)

The input must be a positive value. The displayed distribution represents the probability of the maximum or minimum value corresponding to a particular probability level given a specified number of samples. For example, in the screen shown above, the distribution represents the uncertainty in the probability level of the maximum value assuming 15 samples. It indicates that the expected value of the maximum probability level is 0.9375.

**Note:** The Extreme Probability distribution must always be dimensionless, as it represents a probability level (otherwise GoldSim will display an error).

The Extreme Value distribution (also referred to as the Gumbel distribution) is a continuous distribution used to describe the maximum (or the minimum) observed values for a variable. For example, you could use this distribution to specify the probability of the maximum flow rate in a river in a particular year by fitting a list of the maximum values observed for each of the past 25 years to this distribution. It is therefore commonly used to predict the chance of extreme events such as earthquakes or floods.
It is defined by specifying two parameters: a **Location parameter** and a **Scale parameter**. You must also specify whether you are interested in the **Maximum value** or the **Minimum value**.

**Gamma Distribution**

The Gamma distribution is similar to a Log-Normal, but is less positively-skewed. Because it mathematically represents the time required for the occurrence of a specified number of Poisson events, it is often used to represent the time required to complete a particular task or reach a particular milestone. It is defined by a **Mean** and a **Standard Deviation**:

If desired, a Gamma distribution can be **Truncated**, in which case you must specify a **Minimum** and **Maximum** value. Note that in this case, the Mean and Standard Deviation inputs refer to the statistics prior to truncation.
Note: If the Mean and Standard Deviation are identical, the Gamma distribution collapses to an exponential distribution.

Log-Normal Distribution

The Log-Normal distribution is used to describe quantities in which the logarithm of the value is normally distributed. The Log-Normal is typically used to represent physical quantities which must be non-negative and are positively skewed. This distribution is often used to represent quantities with large (order-of-magnitude) uncertainties.

Log-Normal distributions can be entered in two ways: 1) by entering the Geometric Mean and the Geometric S.D.; or 2) by entering the True Mean and the True S.D. If Geometric Value is checked, the former method is used. When one option is chosen, the other set of parameters are computed internally by GoldSim and are grayed out.

With regard to a set of data points, the Geometric Mean is equal to the exponentiated mean of the logs (i.e., $10^X$, where $X$ is equal to the mean value of the logarithms of the data points). With regard to a set of data points, the Geometric S.D. is the exponentiated standard deviation of the logs (i.e., $10^Y$, where $Y$ is equal to the standard deviation of the logarithms of the data points). Note that the Geometric S.D. is always dimensionless, and must be greater than 1.

The True Mean and the True S.D. refer to the true (arithmetic) mean and S.D. (i.e., in linear space).

A Log-Normal distribution can be Truncated. Note that in this case, the inputs (i.e., Mean and Standard Deviation) refer to the statistics prior to truncation.

All inputs for a Log-Normal must be positive values.

Negative Binomial Distribution

The Negative Binomial distribution describes the distribution of the number of failures in order to achieve a specified number of successes, with a specified probability of success for each attempt. As such, by definition, it is dimensionless. It is frequently used in actuarial models.
The distribution requires two inputs, the **Number of Successes** (which must be a dimensionless positive integer) and the **Probability of Success**. Like the Poisson distribution, the Negative Binomial is a discrete distribution consisting of only integer values.

![Negative Binomial Distribution](image)

**Note:** Since this is a discrete distribution, the PDF view does not actually display a *probability density function*. Instead, it displays a *probability mass function*.

**Normal Distribution**

The Normal distribution is perhaps the most commonly used probability distribution. It can be used, for example, to represent uncertainties resulting from unbiased measurement error, and (because of the central limit theorem) the value of a sum of other variables.

The Normal distribution is defined by a **Mean** and a **Standard Deviation**.

GoldSim also allows you to truncate the distribution by checking the **Truncated** box and specifying a **Minimum** and a **Maximum**.
Pareto Distribution

The Pareto distribution is a continuous, long-tailed distribution that is commonly used to model traffic patterns in network modeling, the size of insurance claims, and income levels in economic modeling.

It requires two inputs, a shape parameter \( a \) and a scale parameter \( b \), which defines the lower bound and mode of the Pareto distribution. Both of these parameters must be positive numbers.

GoldSim also allows you to truncate the distribution by checking the **Truncated** box and specifying a **Maximum**.

Pearson Type III Distribution

The Pearson Type III distribution is a continuous distribution that is used for representing skewed observations. It is widely used in hydrological applications (often for the log of the underlying variable, such as a flood flow).
It is defined by specifying three parameters: a **Location parameter**, a **Scale parameter**, and a **Shape parameter**:

The Poisson distribution is used to represent the number of occurrences or instances of a particular event (e.g., number of storms during a year, number of defects in a length of a pipe).

It is a discrete distribution containing only integer numbers. It is defined by a single value, the **Expected Value** of the number of occurrences or instances (which must be positive):

**Note:** Since this is a discrete distribution, the PDF view does not actually display a probability density function. Instead, it displays a probability mass function.
The Sampled Results distribution provides a flexible way to create a non-parametric distribution using a list of sampled (observed) results. GoldSim generates a CDF by sorting the observations and assuming that a cumulative probability of $1/(\text{Number of Observations})$ exists between each data point. If there are multiple data points at the same value, a discrete probability equal to $N/(\text{Number of Observations})$ is applied at the value, where $N$ is equal to the number of identical observations.

In addition to providing the list of observations, you also specify whether or not GoldSim should extrapolate at the ends of the data set. If the Extrapolation option is cleared, a discrete probability of $1/(2\times\text{Number of observations})$ is assigned to the minimum and maximum values. When the Extrapolation option is selected, GoldSim extends the generated CDF to cumulative probability levels of 0 and 1 by extrapolating existing observations.

A simple example (using 200 data points) is shown below:

Typically, a Sampled Results distribution would only be used with a fairly large number of data points (50 or more). With fewer data points, in most cases, it would be more appropriate to fit the data to a known parametric distribution.

**Note:** You can copy and paste the entries for a Sampled Results distribution from a spreadsheet. To do so, select the column of data points in the spreadsheet, and copy them to the clipboard. Then place the cursor in the first field in the GoldSim dialog, and paste (by pressing Ctrl+V).

The Student's $t$ distribution is most commonly used to represent the distribution of an uncertain population mean, given a sample mean and standard deviation. It is characterized by one parameter, the number of degrees of freedom, which equals the number of samples minus one. The $t$ distribution is symmetric.

The $t$ distribution is often used as follows:

$$p(\mu > X + t_{c} \frac{s_{x}}{\sqrt{n}}) = c$$
where:
\( \mu \) = the population mean;
\( X \) = the sample mean;
\( t_c \) = the t-distribution for \( n-1 \) degrees of freedom, at a cumulative value of \( c \);
\( s_x \) = the sample standard deviation;
n = the number of samples; and
c = the desired confidence level.

The Triangular distribution is useful in situations where you may have little information about a value, but can provide upper and lower bounds, as well as a "best guess" or most likely value. By default, it is defined by specifying either a Minimum (0th percentile), Most Likely and Maximum (100th percentile):
Alternatively, if you check the 10% / 90% checkbox, you can specify the 10th percentile, Most Likely and 90th percentile.

You can also specify that the logarithms of the values have a triangular distribution (by checking the Log box). In this case, all inputs to the distribution must be positive values. Note that for a Log-Triangular distribution, the Most Likely input represents the value whose logarithm is most likely. Log-Triangular distributions are often applied to quantities with large (order-of-magnitude) uncertainties.

**Uniform Distribution**

The Uniform distribution is useful for defining quantities that vary uniformly between two values. You enter a Minimum and a Maximum:

You can also specify that rather than the values being uniformly distributed, the logarithms of the values are uniformly distributed (by checking the Log box). In this case, the Minimum and Maximum must be positive values. Log-Uniform distributions are often applied to quantities with large (order-of-magnitude) uncertainties.

**Weibull Distribution**

The Weibull distribution is often used to represent the distribution of failure times in reliability models (e.g., the distribution of lifetimes for light bulbs). It is defined by a Minimum, a Slope, and Mean-Minimum. If desired, it can be Truncated at its upper end by defining a Maximum:
Note: The Slope of the Weibull is always dimensionless. If it is specified as 1, the distribution collapses to an exponential distribution.

Controlling When a Stochastic Element is Sampled

By default, Stochastic elements are sampled once per realization (typically at the beginning). Hence, if you specify the parameters describing a Stochastic as a function of time, unlike other GoldSim elements, Stochastic elements are not updated (resampled) if the parameters change.

GoldSim does provide a mechanism, however, to force a Stochastic to either 1) be sampled at a specified time; or 2) be resampled multiple times during a simulation. In particular, you can define one or more triggers for the Stochastic which specify when (or under what circumstances) the Stochastic is resampled.

How a Stochastic is resampled is controlled by the Mode specified on the dialog:

![Stochastic Element Dialog]

The default is for the Stochastic to be Sampled once. However, if Resampled is selected, the Resample… button in the dialog becomes available, and this allows you to force the Stochastic to be resampled (triggered) at other points in the simulation.

Read more: Triggering a Stochastic (page 407).

Note: You cannot specify triggers to resample a Stochastic if it is correlated to a second Stochastic (see below). However, if a Stochastic is correlated to a second Stochastic, and that second Stochastic is resampled, the correlated Stochastic will also automatically be resampled (even though no resampling triggers are specifically defined, and the Mode is set to Sampled once).
Conditional Containers can also be used to control when Stochastic elements are resampled.

Read more: Using Conditional Containers (page 899).

An example file which illustrates how a Stochastic is sampled (BasicStochastic.gsm) is in the General Examples/Stochastic folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

Frequently, parameters describing a system will be correlated (inter-dependent) to some extent. For example, if one were to plot frequency distributions of the height and the weight of the people in an office, there would likely be some degree of positive correlation between the two: taller people would generally also be heavier (although this correlation would not be perfect).

The degree of correlation can be measured using a correlation coefficient, which varies between 1 and -1. A correlation coefficient of 1 or -1 indicates perfect positive or negative correlation, respectively.

A positive correlation indicates that the parameters increase or decrease together. A negative correlation indicates that increasing one parameter decreases the other. A correlation coefficient of 0 indicates no correlation (the parameters are independent of each other).

The preferred method to represent a correlation is to explicitly model the cause of a dependency. For example, the height and weight of children are correlated primarily due to a common dependency on age (both increase with age). In such a situation, the best way to represent this correlation would be to explicitly define the mean and standard deviation of the height and the weight as functions of the primary parameter causing the dependency (age).

In many cases, however, specifying such a dependency may be difficult. Therefore, GoldSim provides a mechanism to specify a statistical correlation.

To do so, you must use the Correlation section of the Stochastic dialog (accessed via the More button). You will note that in that section, GoldSim provides three options for correlation:

None. This is the default. The Stochastic is not correlated.

Correlated to: This allows you to correlate one Stochastic to another Stochastic. You must specify the Element (it must be a Stochastic element), and the Value (the correlation factor). The Value represents a rank correlation coefficient. It must be a value between −1 and 1, inclusive (1 being a perfect positive correlation, and −1 being a perfect negative correlation). It can be entered as a number, a link or an expression.

The algorithm used to implement the correlations is described in Appendix B.
Note: If a Stochastic is correlated to a second Stochastic, and that second Stochastic is resampled, the correlated Stochastic will also automatically be resampled.

Read more: Controlling When a Stochastic Element is Sampled (page 177).

Percentile. This allows you to specify a percentile of the distribution. For example, specifying 0.95 would return the 95% percentile. Among other applications, by using this with multiple elements, you could force a particular correlation scheme. An important point should be noted: when the Correlation is specified in this way, the Stochastic is automatically resampled whenever the specified Percentile or any other input parameter for the distribution is changed.

GoldSim also allows you to autocorrelate a Stochastic element (i.e., correlate a Stochastic to its previously realized value). This is useful, for example, when you are sampling a Stochastic multiple times in a realization (e.g., once a month), and this month's value is correlated to last month's value.

In order to autocorrelate a Stochastic, you must first instruct GoldSim to resample the Stochastic (since autocorrelation only has meaning if a Stochastic is resampled).

Read more: Controlling When a Stochastic Element is Sampled (page 177).

After you do so, you will note that in the Correlation section of the Stochastic dialog, GoldSim provides two options for correlation:

None. This is the default. The Stochastic is not autocorrelated.

Autocorrelated. In this case, the Element is autocorrelated (correlated to its previous value). You must specify the Value (the correlation factor). The Value represents a rank correlation coefficient. It must be a value between –1 and 1, inclusive (1 being a perfect positive correlation, and –1 being a perfect negative correlation). It can be entered as a number, a link or an expression.

If you specify autocorrelation, GoldSim will correlate the next update (sample) of the Stochastic to the previous update of the Stochastic. Hence, if you trigger the Stochastic to be sampled every day, it will correlate today’s value to yesterday’s value.

When using autocorrelation, it is important to remember that the correlation factor (Value) should actually be a function of the resampling period. That is, if you resample every 1 minute, the Value should obviously be larger (closer to 1) than if you resample every day. Therefore, when you define a correlation factor for autocorrelation, you should define it with the resampling period in mind.

If you know the autocorrelation coefficient for one resampling period, you can compute it for a different sampling period using the following equation:
C2 = \exp[\ln(C1) \times (\frac{DT2}{DT1})]

where C1 is the coefficient for sampling period DT1, and C2 is the coefficient for sampling period DT2. Note that if you are resampling every timestep (OnChanged Etime), and need to compute an appropriate value for C2, DT2 should be specified as Timestep\_Length (as it may change during the simulation).

The above discussion relates to correlating Stochastics that are defined as scalars. Stochastics can also be defined as vectors (arrays), and in that case, it is possible to specify a correlation matrix to define how the items of the vector are correlated. Stochastic vectors cannot be autocorrelated. They can be correlated to other vectors (in which chase the correlation is term-by-term) or to scalars.

Read more: Creating a Stochastic Vector (page 183).

An example file which illustrates correlation of Stochastics (Correlation.gsm) is in the General Examples/Stochastic folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

In some cases, you will want to run a deterministic simulation (i.e., a single realization which does not try to represent the uncertainty in inputs), even though the model contains some Stochastic elements. For these kinds of simulations, it is necessary to specify the value that a Stochastic element takes on during the simulation.

In order to run a deterministic simulation, you must open the Simulation Settings dialog, go to the Monte Carlo tab, and select the Deterministic Simulation radio button.

Read more: Deterministic Simulation Options (page 482).

For each Stochastic, you must then define the Deterministic Value to be used when a deterministic simulation is carried out:

To access this portion of the dialog, you may need to press the More button on the Stochastic dialog.

The default is for the mean (expected) value of the Stochastic to be used as the Deterministic Value. By pressing the Define... button, however, you can specify other values:
The other options are to use the median value, use a specified quantile (e.g., 0.95), or to use a specified value (e.g., 3.7 m).

If you select one of the first three options in the dialog, you can choose to apply the selection to all Stochastics in the model (by checking the **Apply to all Stochastic elements** box).

If you select the fourth option (a specified value), you must specify the units when you enter the value.

Within the Simulation Settings dialog, when running a deterministic simulation, you can choose to use these individually specified deterministic values, or override them all and use the mean or a specified quantile.

---

**Note:** When you define a specified value, GoldSim remembers it, so that if you switch to a mean value, a median value or a quantile, and then switch back to a specified value, GoldSim retains that value (and you do not have to enter it again).

---

**Note:** A specified value can be outside the defined range of the distribution.

---

**Applying Importance Sampling to a Stochastic Element**

For risk analyses, it is frequently necessary to evaluate the low-probability, high-consequence end of the distribution of the performance of the system. Because the models for such systems are often complex (and hence need significant computer time to simulate), it can be difficult to use the conventional Monte Carlo approach to evaluate these low-probability, high-consequence outcomes, as this may require excessive numbers of realizations.

To facilitate these type of analyses, GoldSim allows you to utilize an importance sampling algorithm to modify the conventional Monte Carlo approach so that the tails of distributions (which could correspond to high-consequence, low-probability outcomes) are sampled with an enhanced frequency. During the analysis of the results that are generated, the biasing effects of the importance sampling are reversed. The result is high-resolution development of the high-consequence, low-probability “tails” of the consequences, without paying a high computational price.

Importance sampling is specified by selecting an option from the **Importance Sampling** button (the default is “None”):
If you select “High-End”, GoldSim will preferentially bias toward the high end of the distribution; if you select “Low-End”, it will preferentially bias toward the low end of the distribution.

It is important to understand that you should use importance sampling sparingly (i.e., only for those elements that really need it). This is because the degree of biasing for distribution tails that GoldSim can apply decreases with the number of elements for which importance sampling is applied.

The importance sampling algorithm is discussed in detail in Appendix B.

**Note:** Importance sampling cannot be applied if the Stochastic is resampled during a realization.

**Read more:** Controlling When a Stochastic Element is Sampled (page 177).

**Note:** In addition to the importance sampling method described here (in which you can choose to force importance sampling on either the low end or high end of a Stochastic element’s range), GoldSim also provides an advanced feature that supports custom importance sampling that can be applied over user-defined regions of the Stochastic element’s range.

**Read more:** Customized Importance Sampling Using User-Defined Realization Weights (page 1013).

### Specialized Functions That Operate on Distributions

GoldSim provides several specialized functions that can be used to reference the Distribution output type and return properties of the distribution. The Distribution output is a complex output representing all the distribution information. It can be produced by a Stochastic element, a Spreadsheet element, or a SubModel.

These functions all require as the first argument a Distribution output. Some require an additional output:

- **PDF_Value(Distribution, P):** The value of the distribution at the specified cumulative probability level P. The output has the same dimensions as the distribution being referenced.

- **PDF_CumProb(Distribution, V):** The cumulative probability of the distribution at the specified value V. The output is dimensionless.

- **PDF_Mean(Distribution):** The mean of the distribution. The output has the same dimensions as the distribution being referenced.
PDF_SD(Distribution): The standard deviation of the distribution. The output has the same dimensions as the distribution being referenced.

PDF_CTE(Distribution, P): The conditional tail expectation of cumulative probability P (i.e., the expected value of the output given that it lies above a specified cumulative probability P). This result represents the mean of the greatest 100(1 - P)% of outcomes. The output has the same dimensions as the distribution being referenced.

Note: Calculation of the Conditional Tail Expectation is discussed in detail in Appendix B.

PDF_Density(Distribution, V): The probability density or discrete probability of the distribution at the specified value V. If the distribution is continuous, the result is the probability density of the specified value. The dimensions are the inverse of the distribution itself. If the distribution is discrete, then result is the discrete probability of the specified value, in which case it is dimensionless.

Note: For Boolean distributions, the value argument V must be a condition. Note, however, that if a condition-type result is passed out through a SubModel interface, it must be either 1 or 0.

Note: In some cases, GoldSim cannot compute a direct solution for a statistic (e.g., computing the standard deviation for a truncated Weibull). In this case, the function returns an error.

Since these functions simply return the key properties of a probability distribution, in most cases, there would be no value in using these functions directly on Stochastic elements. However, there are two primary applications for these functions:

- When the definition of a Stochastic distribution is changing dynamically during a simulation (either by making the defining parameters functions of time, or via simulated Bayesian updating), these functions can be used to query the current shape and characteristics of the distribution.

  Read more: Dynamically Revising Distributions Using Simulated Bayesian Updating (page 1016).

- When exporting a distribution result from a Monte Carlo SubModel, these functions can be used to query the current shape and characteristics of the distribution in the parent model.

  Read more: Creating the Output Interface to a SubModel (page 984).

Creating a Stochastic Vector

GoldSim allows you to define a Stochastic element as a vector of data. If you define your element as a vector, rather than inputting a single probability distribution, you specify a set of distribution parameters (one for each item of the vector).

Read more: Using Vectors and Matrices (page 782).
To define your element as a vector, press the **Type...** button on the dialog. The following dialog will appear:

From the **Order** drop-list select “Vector (1-D Array)”. You will then need to specify a set of **Row Labels**.

**Read more:** [Understanding Array Labels](page 783).

If you specify the Stochastic as a vector, the dialog for defining the distribution (accessed via the **Edit Distribution...** button) will look similar to this:

This dialog is identical to the standard dialog for a scalar Stochastic, with two exceptions:

- The input fields for the parameters for the distribution must be vectors. Hence, when you create a Stochastic vector, all of the items of the vector have the same distribution type, but may have different values for the input parameters for the distribution.
- You view the distribution and the statistics for a particular item of the vector by selecting the item from the drop-down list at the top of the dialog (immediately to the right of the distribution type).

**Note:** If the distribution type is Cumulative, Discrete or Sampled Results, you cannot enter vectors into the grids defining the distribution. That is, the grids only accept numbers. As a result, all items of the vector would be identical. If you want to create a vector of Stochastics that consists of Cumulative, Discrete or Sampled Results distributions, you can do so by creating a separate Stochastic for each item, and then using a Data element (defined as a vector) to reference each separate item.
A Stochastic vector cannot be autocorrelated. It can, however, be correlated to another Stochastic. If the other Stochastic is a vector, the correlation is term-by-term. If the other Stochastic is a scalar, all terms are correlate to the scalar value.

You can also specify correlations between members of the vector via a correlation matrix. A correlation matrix specifies the correlations between variables, and generally has the following form:

```
<table>
<thead>
<tr>
<th></th>
<th>Alpha</th>
<th>Beta</th>
<th>Gamma</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Beta</td>
<td>0.1</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>Gamma</td>
<td>0.1</td>
<td>0.99</td>
<td>1</td>
</tr>
</tbody>
</table>
```

Note that by definition, a correlation matrix is symmetric around its diagonal (since the cross diagonal terms define the same correlation coefficient).

If you specify that a Stochastic is a vector, then the drop-list in the Correlation section of the dialog provides an option called “Matrix”. If this option is selected, a button is provided (Define…) to define the correlation matrix:

This button provides access to a dialog for specifying the correlation matrix:

![Correlation Matrix Dialog](image)

As pointed out above, by default, all off-diagonal correlation coefficients are zero. The matrix is symmetrical, so you need only define one of the cross-diagonal terms. The value represents a rank correlation coefficient, and must vary between -1 and 1. It must be a number (i.e., you cannot specify a link).

**Note:** When you define a correlation matrix, it is important to ensure that it is internally consistent. For example, if you specified that A was positively correlated to B, and B was positively correlated to C, but that A was negatively correlated to C, the correlation matrix would be inconsistent (since in this case, A should also be positively correlated to C). When this occurs, GoldSim will produce a fatal error message.
GoldSim provides several different algorithms for correlating the members of the vector. These are selected from the Correlation Algorithm drop-list at the top of the dialog. The various correlations algorithms are discussed in detail in Appendix B.

The browser view of a Stochastic element is shown below:

Note: Element inputs and outputs are only shown in the browser if you choose to Show Element Subitems (accessed via the browser context menu by right-clicking in the browser).

All Stochastic elements have four outputs.

Note that the Distribution output has a different icon than the other outputs. This is because the Distribution output is a complex output that represents all the statistical information necessary to define a probability distribution. It can only be used in several specialized locations.

The number of inputs is a function of the distribution type. If the Stochastic is correlated, it has a Correlated To input, as well as a Factor input (the correlation factor).

Note: If you have specified a quantile or a specified value for the Deterministic Value, this also appears as an input in the browser. In addition, if the Stochastic is explicitly triggered, a number of additional inputs related to triggering also appear.

Using Time Series Elements

Time Series elements are a type of input element that provide a very flexible and powerful way to input external time histories of data into GoldSim (i.e., a table showing how a variable changes as a function of time).

You can use a Time Series element to input the predefined histories of things like interest rates, commodity prices, temperature, rainfall rates, sales rates, cash flows, or the condition (reliability, strength) of a machine or a material. A Time Series element represents an exogenous (external) “driver” that can directly or indirectly influence the dynamic behavior of the system being modeled.

Note: Time Series elements are for time series that are known in advance of the simulation. If you want to generate random (stochastic) histories during your simulation, you should use a History Generator element instead.
Read more: Generating Stochastic Time Histories (page 826).

To define a Time Series element, you input a set of time series records (i.e., a table of times and values) and specify what the data represent (e.g., instantaneously measured values at specified times). Note that the time values in the input time series do not have to coincide with the model’s timesteps.

You then specify what the output of the element should be (e.g., the average value of the variable over each timestep). GoldSim then carries out the appropriate calculations in order to convert the input data into the specified output.

For example, if the input data is specified as a series of instantaneous values, GoldSim interpolates between the time series records to produce the output time series.

The dialog for a Time Series element looks like this:

Note: A number of advanced options are initially hidden by the More button.

There are five steps involved in defining a Time Series element:

1. Define the type (scalar, vector, or matrix; value or discrete change) and units for the time series;
2. Specify the source for the input data (e.g., defined directly, imported from spreadsheet);
3. Specify what the time series input represents (e.g., instantaneous values at specific times, constant values over specified intervals);
4. Define the time series itself (if specified directly) or the location of the data source (if imported); and
5. Specify the output types to be computed for the element (e.g., instantaneous values at each timestep, average values over a timestep).

Each of these steps, as well as some example files and advanced options, are described below.
Note: In addition to inputting pre-defined histories, one of the advanced options for a Time Series element is to use it to record (and subsequently play back) a history computed by another GoldSim element.

**Defining the Data Type and Units for a Time Series**

Read more: Using a Time Series Element to Record and Play Back a History (page 221).

Several files are provided within the TimeSeries subfolder of the General Examples folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) that include examples on the use of Time Series elements.

The first step involved in defining a Time Series element is to define the data type and units for the time series.

Units are specified in the Display Units field. This defines the dimensions of the input and primary output. By default, a Time Series is dimensionless.

Like all GoldSim elements, the data type is specified by pressing the Type... button.

In most applications, your time series will represent a set of scalar values (one value at each specified time point). Because this is the default, in most situations, there is no need to edit the Type.

If you do press the Type... button, the following dialog is displayed:

![Define output attributes of Time Series element](image)

The **Type** can be defined as a Value (the default) or a series of Discrete Change Signals.

Read more: Generating Discrete Changes Using Time Series Elements (page 415).

The **Order** can be defined as a Scalar (the default), a vector or a matrix.

Defining time series of vectors or matrices (i.e., in which each time point is defined in terms of an array of data, rather than just a single datum) is useful in some situations (e.g., defining a time series for the concentrations of a set of chemicals, or the prices for a set of securities).

Read more: Specifying Time Series Data as Vectors or Matrices (page 199).

After the data type and units are defined, the next step is to define the source of the input data for the Time Series.

Specifying the Source of the Input Data for a Time Series

Time Series elements can be defined in four ways. The two most common ways to define a Time Series are as follows:

- You can define the series locally by entering the data directly; or
- You can import the data by linking to a spreadsheet.
In addition, GoldSim provides two advanced options for defining Time Series elements:

- You can read the output of any other element in GoldSim, and "record" the results, and then "play them back" in a subsequent run of the model. Among other applications, this allows you to use the results from one model as input to another model.

- You can link to a "time series definition" that was output by another Time Series element within an embedded SubModel or from an external function (a DLL). The primary application of this advanced option is to transfer time series between SubModels.

Read more: Using SubModels to Embed Models Within Models (page 973).

The way that the input data are defined is specified in the Data Source drop-list:

- **Locally defined data.** This is the default. You enter the data directly into a table using the Edit Data button.

Read more: Viewing and Editing Time Series Inputs (page 191).

- **Imported from MS-Excel Spreadsheet.** You import the data from an Excel spreadsheet. When you select this option, a new tab appears (Excel) from which you can specify the spreadsheet filename and other required information. You can import the data immediately, or wait until the simulation begins (the data will always be automatically imported when you start a simulation). The View Data button allows you to view the imported data (it cannot be edited unless you switch back to "Locally defined data").

Read more: Importing Data into a Time Series from a Spreadsheet (page 194).

- **Linked to external Time Series Definition.** In this advanced option, you read a "time series definition" that was output by another Time Series element (within a separate SubModel) or from an external element. When you select this option, a new tab appears (Linked) from which you specify the link from the other element.

Read more: Referencing a Time Series Definition Output (page 225).

- **Recording output history.** In this advanced option, you can select an output of any other element in GoldSim, and "record" the results (in order to "play them back" in a subsequent run of the model). When you select this option, a new tab appears (Recording) from which you specify the link from the other element, as well as several additional options regarding how the output is recorded.

Read more: Using a Time Series Element to Record and Play Back a History (page 221).

After specifying the source of the input data for the Time Series, the next step is to specify what the input to the Time Series represents.
The input to a Time Series element is always a set of time series records (i.e., a table consisting of time/value pairs). However, since a time series can represent several different kinds of data (e.g., values, rates), the first (and most important) step in defining a Time Series element (prior to actually defining the records) is to specify what these records represent. GoldSim provides several different ways for you to define the inputs to Time Series elements.

In particular, you can specify the input to a Time Series as representing one of six kinds of data:

- **The instantaneous value** of a variable at specified times: Examples include the price of a commodity (e.g., gold), the price of an asset (e.g., a stock), or the height of a projectile.

  Read more: [Example: Time Series Inputs Represent Instantaneous Values](page 204).

- **The constant value** of a variable over the next time interval or the constant value of a variable over the previous time interval. In some cases, a value is assumed to be constant over a specified interval (i.e., follow a "stair-step" function from interval to interval). In some instances, this may reflect the fact that the variable actually is held constant over each interval. More commonly, it may be an artifact of how the data was recorded (e.g., it may have only been recorded in terms of an average value over the interval). GoldSim provides two options for entering this kind of data (one is forward-looking, and one is backward-looking).

  Read more: [Example: Time Series Inputs Represent Constant Values Over Specified Intervals](page 206).

- The (uniform) change (increment) in a variable over the next time interval or the (uniform) change (increment) in a variable over the previous time interval: These options are usually used to output cumulative values (e.g., cumulative sales, cumulative rainfall) and/or rates (e.g., sales rates, rainfall rates) that are specified by defining incremental changes in a variable (e.g., monthly sales, monthly rainfall) over specified intervals. When this option is selected, the Primary Output of the Time Series is not the specified time series itself; rather, it is the cumulative value of the variable at each point in the simulation (assuming that the specified changes occur uniformly over each interval).

  Good examples of this type of time series data are rainfall and sales rates. In these cases, what is actually measured and recorded are totals (e.g., rainfall in mm or sales in $) over a particular interval (e.g., 31 days). Based on this data, you could then use GoldSim to compute the cumulative rainfall and sales (over the entire simulation), as well as the average daily rainfall or rate for each interval (e.g., month). GoldSim assumes that the variable (e.g., cumulative rainfall or sales) changes uniformly over the time period, which means that the rate of change is constant over the interval (i.e., it "stair-steps"). GoldSim provides two options for entering this kind of data (one is forward-looking, and one is backward-looking).

  Read more: [Example: Time Series Inputs Represent Changes Over Specified Intervals](page 209).

- **Discrete changes** to a variable at specified times. In some cases, your time series may not represent the current value of a variable, but a series of discrete changes (additions and subtractions) to a variable. An example of this kind of time series deposits and withdrawals from an account that
Using Time Series Elements

happen discretely. Note that this option is only available if the Data Type has been defined as a Discrete Change Signal with an Add instruction.

**Read more:** Example: Time Series Inputs Represent Discrete Changes (page 211); Defining the Data Type and Units for a Time Series (page 188).

Prior to defining the time series records, you must select one of these four options from the drop-list labeled **Represents**.

After specifying what the input to the Time Series represents, the next step is to define the actual Time Series input data.

The data for your time series are viewed and edited by pressing the **Edit Data...** (or, in some cases, the **View Data...** button) in the main Time Series dialog.

Note: Time series data can only be edited if the **Data Source** is defined as "Locally defined data". If any other Data Source is specified, the data can only be viewed (and cannot be directly edited) in this dialog.

When you press the **Edit Data...** (or **View Data...**) button, the following dialog is displayed:

You can enter an optional **Description** at the bottom of the dialog to help document the time series data.

If you are defining the data locally, you can add rows to the table using the **Add Row** button. If you hold down the **Ctrl** key when you press this button, you will be prompted for the number of rows to add (otherwise, a single row will be added).

You can also add rows by pasting in a block of data from the Windows clipboard.
To delete one or more rows, select them and press the **Remove Row** button. If you hold down the Shift key when you press this button, all of the rows from the selected row to the bottom will be deleted.

All shortcut keys for editing the time series table are summarized below:

<table>
<thead>
<tr>
<th>Shortcut Key</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl+A</td>
<td>Selects all rows</td>
</tr>
<tr>
<td>Arrow keys</td>
<td>Move active cell up, down, left or right</td>
</tr>
<tr>
<td>Ctrl+Left arrow</td>
<td>Move active cell to first column</td>
</tr>
<tr>
<td>Ctrl+Right arrow</td>
<td>Move active cell to last column</td>
</tr>
<tr>
<td>Ctrl+Up arrow</td>
<td>Move active cell to first row</td>
</tr>
<tr>
<td>Ctrl+Down arrow</td>
<td>Move active cell to last row</td>
</tr>
<tr>
<td>Ctrl+Insert N</td>
<td>Insert N new rows below active cell</td>
</tr>
<tr>
<td>Shift+Delete</td>
<td>Delete all rows from active cell (or first selected row) to bottom</td>
</tr>
<tr>
<td>Ctrl+Tab or Ctrl+Insert</td>
<td>Insert new row below active cell</td>
</tr>
<tr>
<td>Shift+Ctrl+Tab</td>
<td>Insert new row above active cell</td>
</tr>
<tr>
<td>Ctrl+Backspace or Ctrl+Delete</td>
<td>Delete row containing active cell or all currently selected rows</td>
</tr>
</tbody>
</table>

**Note:** When you press **OK** to close the dialog, GoldSim deletes all blank rows (rows in which the time value and data value are not defined).

**Note:** You can select one of the columns by clicking on the columns header. You can select both columns by clicking on the first column header and dragging to the second. You can select the entire table (including headers and row numbers) by clicking in the upper right-hand corner of the table.

The fields in the table only accept numbers (they do not accept links). Moreover, you should not append units to the numbers.

The time entries can be entered either as numbers or as dates. If your Simulation Settings are set to Elapsed Time when you insert the element, they default to being entered as numbers; if your Simulation Settings are set to Date-Time when you insert the element, they default to being entered as dates. However, you can override the default setting by changing the selection in the **Time Unit** field.

**Read more:** [Entering Time Series Data as Dates or Elapsed Times](page 197).

If time entries are entered as numbers, the assumed units for the time column are defined in the **Time unit** field. (This field defaults to the Time Display Units specified in the **Time** tab of Simulation Settings dialog.)

You can enter time in seconds, minutes, hours, days, weeks, months or years:
Note: Because month and year do not represent a constant period of time (the length of these units changes depending on the month or the year), GoldSim assumes that each of these units is defined by their average value. In particular, one year is assumed to consist of 365.25 days, and one month is assumed to consist of 30.4375 days (365.25/12).

If you select "Calendar Time" from the Specify time unit field, time entries are entered as dates. The date format that is expected is determined by the Windows settings for your machine. A typical setting might be MM/DD/YYYY HH:MM:SS. (To change these settings, go to Control Panel and edit the Regional Options.)

The assumed units for the data are the Display Units specified on the main Time Series dialog. The assumed units are displayed in the column headers.

Note: The values of Time must increase monotonically as you move downward in the table. If they do not, GoldSim will automatically sort the rows into the correct order.

Two important rules must be followed when defining time series data:

- Unless your input data represent Discrete Changes, the first data point must be at or prior to the beginning of the simulation. That is, if you have specified time entries as elapsed times, the first data point must be 0. If you have specified time entries as dates, the first date must be at or prior to the start time for the simulation.

- Unless you are using the Time Shifting feature, if the data is specified to represent either "Instantaneous value" or "Change over the next time interval", the last data point must be at or beyond the end of the simulation. That is, if you have specified time entries as elapsed times, the last data point must be greater than the simulation duration. If you have specified time entries as dates, the last date must be at or after the end time for the simulation.

Read more: Time Shifting Time Series Data (page 214).

If either one of these requirements is not met, GoldSim will issue a fatal error when you try to run the model.

After you have defined the data and closed the dialog, the number of defined data points is indicated in the main Time Series dialog.
You can link a Time Series element directly to a spreadsheet. This allows you to automatically import data from a spreadsheet at the beginning of each simulation.

You link a Time Series element to a spreadsheet by selecting “Imported from MS-Excel Spreadsheet” from the Data Source field in the Data Definition portion of the dialog:

When you do so, a new tab (Excel) is added to the dialog that allows you to define the properties of the spreadsheet link:

You must first enter the name of a Microsoft Excel spreadsheet file by pressing the Options >> button. This will provide options for either selecting an existing file, or creating (and then selecting) a new file.

Note: If you select a file in the same directory as (or a subdirectory below) your GoldSim .gsm file, GoldSim will subsequently display just a local path. If you select a file in a directory above your .gsm file, it will display the full path.

Once you have selected a file, you can subsequently use the Options >> button to select a different file. You can also use the Options>> button to open the selected file in Excel.
Note: GoldSim supports both .xlsx and .xls Excel files. However, if you have an older version of Excel (prior to Office 2007), you will need to install Microsoft’s Office Compatibility Pack in order to read .xlsx files. Excel 2007 and later support an extended worksheet size (1,048,576 rows by 16,384 columns) than earlier versions (65,536 rows by 256 columns). If you wish to import data from an extended worksheet range into GoldSim, you must use Excel 2007 or newer, and the file format must be .xlsx. Note that GoldSim does not officially support versions of Excel prior to Excel 2003.

Warning: You should never try to open a spreadsheet that is linked to GoldSim while a simulation is running. In addition, prior to running a GoldSim model that is linked to a spreadsheet, all referenced spreadsheets should be closed. If you try to interact with a spreadsheet that is being used by GoldSim during a simulation, all interaction with the spreadsheet is blocked by GoldSim. However, under some circumstances this may not be possible and could lead to errors. Hence, as a general rule, you should not use Excel while a GoldSim model that references Excel is running.

Warning: Within Excel, a date is stored internally as the number of days (which can be fractional) since December 31, 1899 00:00:00. When GoldSim imports a date from Excel, it converts this Julian value to a date. Unfortunately, Excel mistakenly adds an extra day into its calendar that did not actually exist (February 29, 1900). As a result, the effective Julian date reference for all times after March 1, 1900 in Excel is actually December 30, 1899 00:00:00. Within GoldSim, the Run Property Time represents the elapsed time since the reference date December 30, 1899 00:00:00. Hence, for all dates after March 1, 1900, Excel and GoldSim have the same effective Julian date reference. However, for dates prior to March 1, 1900, GoldSim and Excel will differ by one day.

After selecting the spreadsheet file, you must specify the direction of the data in the spreadsheet. That is, the data can be organized into Columns or Rows. Columns implies that the data exists in the spreadsheet vertically: there is one column containing all of the time entries with a different time entry for each row, and another column containing all the value entries with a different value entry for each row. Rows implies that the data exists in the spreadsheet horizontally: there is one row containing all of the time entries with a different time entry for each column, and another row containing all the value entries with a different value entry for each column.

The time and value data do not need to be contiguous rows or columns in the spreadsheet. They must, however, either both exist as rows, or both exist as columns.

You must also specify when GoldSim should stop importing the data. There are two options:

Read data until first empty time value cell is found. Data is read until an empty or non-numeric time value cell is encountered.

Read specific number of rows or columns. A specific number of time value entries are read.
You then specify the location of the data. You do this by specifying the **Excel Sheet** and the **Start Cell** (e.g., A2) for both the series of Elapsed Times and for the series of Values. You can do this directly by typing this information into the **Starting Cell** field. Alternatively, you can press the Edit Location button (located at the end of each row in the dialog), which opens the spreadsheet and provides a dialog allowing you to directly select the desired cell.

**Note:** GoldSim allows you to define a time series of vector (or matrix) data and import it from the spreadsheet. If you define your element as a vector (or matrix), rather than specifying an **Excel Sheet** and a **Start Cell** for a single data value at each time point, you specify these for multiple data values (one for each item of the vector or matrix).

**Read more:** [Specifying Time Series Data as Vectors or Matrices](page 199).

**Note:** The time column must increase monotonically in the spreadsheet. If it does not, GoldSim will stop the import just prior to the first time data point that is not monotonically increasing.

The row or column of time entries that you import from the spreadsheet can exist either as numbers or as dates in the spreadsheet. If they are formatted as dates in the spreadsheet, they will always be imported as dates into the Time Series element. If they are formatted as numbers in the spreadsheet, they will always be imported as elapsed time values into the Time Series element. Once they are imported into the Time Series element, GoldSim provides a mechanism by which you can convert dates to elapsed times, and vice-versa.

**Read more:** [Entering Time Series Data as Dates or Elapsed Times](page 197).

You must also specify the units in which the time and value entries are represented in the spreadsheet. Obviously, the time entries must have dimensions of time. The units for the value entries must have the same dimension as the **Display Units** specified in the Time Series dialog (and defaults to those units).

**Read more:** [Setting the Basic Time Options](page 453).

GoldSim automatically imports data from the spreadsheet at the start of a simulation if either 1) any of the properties on the **Excel** tab have been modified; or 2) the Excel file itself has been changed since the last import. You can also import data manually at any time prior to running a simulation (e.g., so you can view it) by pressing the **Import Data** button in the Excel tab.

**Note:** If GoldSim cannot find the spreadsheet when you try to run the model but you have already imported the data previously, it will use the data that was imported previously and will log a warning message to the Run Log.

**Warning:** You should never try to open a spreadsheet that is linked to GoldSim while a simulation is running. In addition, prior to running a GoldSim model that is linked to a spreadsheet, all referenced spreadsheets should be closed.
If you want to ensure that you do not import data from a spreadsheet that has been inadvertently edited since you last imported the data, you can choose to “Lock onto” a spreadsheet file (by checking “Lock onto selected file” from the Options>> button). If you are locked onto a file, GoldSim will not allow the simulation to run if the file has been modified in any way (the file is set to read only when it is locked onto). In order to run a simulation with a changed file, you must first remove the lock (by clearing “Lock onto selected file” from the Options>> button).

The example file (ImportTimeSeries.gsm) in the TimeSeries subfolder of the General Examples folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) includes an example of how a Time Series can be imported from a spreadsheet.

Note: When you link a Time Series to a spreadsheet, the Edit Data… button changes to View Data…. If you press this button to access the dialog displaying the time series data, you will note that they are no longer editable. Once you are linked to a spreadsheet in this way, you can not edit the data manually unless you change Data Source back to “Locally defined data”. When you do so, GoldSim keeps the imported data and makes it editable.

Note: Spreadsheets cannot be used to import multiple time series. That is, if a Time Series is linked to a spreadsheet, it can only import a single series. Importing multiple time series from a spreadsheet into a Time Series element is not supported.

Pasting Data into a Time Series

You can paste directly from a spreadsheet, Word table or from a comma delimited text file into the table of time series data. To paste data from the Windows clipboard into a table, simply click once in the cell representing the upper left-hand corner of the region of the table into which you wish to paste the data, and press Ctrl+V.

When you paste data into a time series table, GoldSim will overwrite any data existing in the target region, and if necessary, will automatically expand the size of the table (i.e., add rows) to accommodate all of the data being pasted.

You can carry out dynamic simulations in one of two ways in GoldSim (this is controlled via the Simulation Settings dialog):

- **Elapsed Time**: In an Elapsed Time simulation (the default), you specify a Duration and the number of timesteps (and optionally, a Start date-time). The simulation is tracked in terms of elapsed time from the point the simulation began.

- **Date-time**: In a Date-time simulation, you enter a Start date-time and an End date-time, along with the number of timesteps, and the simulation is tracked in terms of the date-time.

Read more: Setting the Basic Time Options (page 453).

Whether you are carrying out a Date-time simulation or an Elapsed Time simulation, you can choose whether your time series data is defined in terms of elapsed time or date-time. If your Simulation Settings are set to Elapsed Time...
when you insert the element, they default to being entered as numbers (with the default **Time Unit** being the Time Display Units specified in the **Time** tab of the Simulation Settings dialog); if your Simulation Settings are set to Date-Time when you insert the element, they default to being entered as dates.

However, you can override the default setting by changing the selection in the **Time Unit** field:

![Edit Time Series Data](image1)

If you select "Calendar Time" from this list, the column expects a date-time to be entered:

![Edit Time Series Data](image2)

**Warning:** If you are importing your time series from a spreadsheet, and the time series is formatted as dates in the spreadsheet, it will always be imported as dates into the Time Series element (even if **Specified time unit** is not set to “Calendar Time”).

**Read more:** [Importing Data into a Time Series from a Spreadsheet](page 194). The date format that is expected is determined by the Windows settings for your machine. To change these settings, go to Control Panel and edit the Regional Options. In the example above, the date-time is specified as MM/DD/YYYY HH:MM:SS. You can use 12 hour formatting (and must then specify AM or PM) or use 24 hour formatting. Note that if you omit the hours, minutes, and seconds, GoldSim assumes 00:00:00 AM (the start of the day).

**Note:** Time Series elements accept dates between 1 January 100 and 31 December 9999.

**Warning:** Dates are entered in expressions in a different way than they are entered into a Time Series element. In particular, in an expression, dates must be surrounded by quotation marks. This requirement, however, does not apply when entering dates into Time Series elements.
Read more: Referencing Dates in Expressions (page 131).

After entering (or importing) time series in one format or the other, you can modify the format changing the selection for Specify time unit. If the data are in Date-time, they will be converted to elapsed time relative to the specified Start-time in the Simulation Settings dialog. If the data are in elapsed time, they will be converted to Date-time by adding the elapsed time to the specified Start-time in the Simulation Settings dialog.

Note: Even when running an Elapsed Time simulation, the Start-time is available for editing in the Simulation Settings dialog.

Specifying Time Series Data as Vectors or Matrices

GoldSim allows you to define a time series of vector or matrix data. If you define your element as a vector or matrix, rather than inputting a single data value for each time point, you specify a set of data values (one for each item of the vector or matrix) at each time point.

Read more: Using Vectors and Matrices (page 782).

To define your element as a vector, press the Type… button on the dialog. The following dialog will appear:

From the Order drop-list select “Vector (1-D Array)”. You will then need to specify a set of Row Labels.

Read more: Understanding Array Labels (page 783).

If you specify the time series as a vector, the dialog for defining the data records for your time series (accessed via the Edit Data… button) will look similar to this:

There is one data column for each item of the vector. In this example, the vector has three items (Gold, Silver, Platinum). Note that each data column is labeled with the vector’s array labels.

To define your element as a matrix, select “Matrix (2-D Array)” as the Order when defining the Type. You will then need to specify a set of Row Labels and a set of Column Labels.
If you specify the time series as a matrix, the dialog for defining the data records for your time series (accessed via the Edit Data… button) will look similar to this:

![Edit Time Series Data dialog]

In this case, a scroll bar at the bottom of the dialog allows you to scroll to see all of the columns.

There is one data column for each item of the matrix. In this example, the matrix has three rows (Gold, Silver, Platinum) and 4 columns (NE, SE, SW, NW). Hence, there are 12 items (12 data columns). Note that each data column is labeled with the matrix's array labels (row, column).

The example file (ArrayTimeSeries.gsm) in the TimeSeries subfolder of the General Examples folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) includes an example of how a Time Series can be defined as a vector or a matrix.

Specifying Time Series Outputs

After you have defined the time series data (the input), it is necessary to define what outputs are to be produced by the element. That is, you must specify how GoldSim should manipulate the time series input data to produce one or more time series outputs that can subsequently be used in your model.

By default, GoldSim produces a single output, which (if the output is a value, as opposed to a discrete change) corresponds to the instantaneous value of the output at each point in time (i.e., at each model timestep), computed by simply interpolating between the specified data points. In many cases, this will meet your needs, and there is no reason to modify this.

However, in some cases, you may want to take advantage of some advanced features of the Time Series element when specifying Time Series outputs. To access these features, you must press the More button on the Time Series dialog:
Pressing this button provides access to the following section of the dialog:

The first two options in this section pertain to options for specifying Time Series outputs.

Because the time series data can be specified at arbitrary time points (that are unrelated to the timesteps used in the GoldSim model), GoldSim allows you to specify exactly how the time series will be mapped to timesteps in GoldSim. GoldSim provides two options for the Primary Output:

If **Instantaneous value** is selected (the default), GoldSim computes the instantaneous value of the output at each point in time (i.e., at each timestep).

If **Average value over the next timestep** is selected, GoldSim averages the output over the projected timestep length. This option should be selected when your time series represents flows or fluxes, as by doing so, you ensure that the flow or flux is properly conserved.
**Note:** If your input data represents Discrete Changes (i.e., the data Type has been defined as a Discrete Change Signal, rather than a Value), the only option available for the Primary Output is "Discrete Change". In this case, the element will generate Discrete Change Signals with an Add instruction at the specified time points (and independent of the scheduled timesteps).

**Read more:** Example: Time Series Inputs Represent Discrete Changes (page 211); Defining the Data Type and Units for a Time Series (page 188).

In addition to the primary output, you can also choose to produce a secondary output that represents the Rate of Change of the variable. This is done by checking the **Enable Rate of Change output** option:

If this box is checked, the element computes a secondary output (called Rate_of_Change). It represents the rate of change of the variable over the next projected timestep. The display units for this output are the display units for the Time Series divided by the Time Display Unit defined in the Simulation Settings dialog.

The Rate_of_Change output is valuable when the data you have collected represents changes in the cumulative value of a variable (e.g., cumulative sales, cumulative rainfall) over specified intervals, and you are interested in computing average rates between the measurement points (e.g., sales rates, rainfall rates). In these cases, you enter changes in the cumulative value of a variable between the measurement points (e.g., every week or every month, or at irregular intervals), and specify that the data represents a "Change over the next time interval". You then specify that you want to "Enable the Rate of Change output". The Rate_of_Change output then represents the average rate (e.g., rainfall or sales rate) for each interval (e.g., month).

**Note:** Because GoldSim computes the **Average Value over the next timestep** primary output and the Rate_of_Change secondary output by assuming a timestep length, and under some circumstances the timestep length could subsequently be shortened (due to an event), the values that are computed could contain a slight error. In such a case, GoldSim tracks any error that is incurred, and corrects for it at the next timestep (in order to ensure conservation of the variable).

**Note:** If the input represents an Instantaneous value, and the last data point corresponds exactly with the end of the simulation, the Rate_of_Change output drops to zero at the last timestep (since GoldSim cannot compute a "forward looking" rate of change at that point).
Referencing a Time Series Using a Function

In some situations, you might want to reference a value from a Time Series element at some point in the future or past (as opposed to at the current time in the simulation). To support this, Time Series elements can be referenced in the way that you would reference a built-in function (like sin or min). That is, once you define a Time Series, you can reference it in input expressions for other elements as if it were a function:

![Equation]

This expression instructs GoldSim to use the Time Series defined by the element named Rainfall and compute its value at time = 35 days. When specifying a date, you must enclose it in quotation marks. The format for referencing a date is determined by the Regional and Language time/date settings specified by the operating system.

Read more: Referencing Dates in Expressions (page 131).

Of course, once you have defined the table, you can reference it at multiple locations (i.e., in the input fields for various elements) using different input arguments (different time values) in the same manner as you would use the built-in functions provided by GoldSim multiple times.

Note that when you create a Time Series, GoldSim automatically lists it as a function in the Function menu that is accessible from the context menu within an input field:

![Function Menu]

The dimensions of the Time Series function are determined by the Display Units specified when the Time Series element was defined. Note that if the Time Series itself is defined as an array, the output of the function is an equivalent array.

Time Series functions have two arguments (with the second one being optional). In most cases, you will call the function with only the first argument. The first argument is the time value. If you omit the second argument, the first argument is interpreted in the same manner in which the Time Series was defined. Hence, if the Time Series was defined based on Calendar Time, the first argument is assumed to be a date. If the Time Series was defined based on Elapsed time, the first argument is assumed to be an elapsed time.

By using the second argument, you can specifically override the default setting for how the first argument is interpreted. In particular, if

Second Argument = 1: The first argument is interpreted as a date.
Using Time Series Elements

Second Argument = 2: The first argument is interpreted as an elapsed time from the beginning of the simulation.

**Warning:** To utilize this function properly, it is important to understand how dates and elapsed times are represented in GoldSim. In particular, note that dates are represented internally in GoldSim as times. In particular, a date is stored as the amount of time since 30 December 1899. As a result, if you specify a first argument of 90 days, but indicate that this is a date (e.g., by specifying the second argument as 1), it will be interpreted as the calendar date 30 March 1900. Similarly, if you specify a first argument of “16 October 2011” and indicate that this is an elapsed time (e.g., by specifying the second argument as 2), it will be interpreted as the elapsed time of 40832 days (the number of days since 30 December 1899).

**Read more:** Referencing Dates in Expressions (page 131).

Note that if the first argument represents a time that precedes the earliest entry in the Time Series, the function returns the value associated with the first entry. Similarly, if the first argument represents a time that is later than the last entry in the Time Series, the function returns the value associated with the last entry.

**Note:** If your Time Series is defined as an array, you can use GoldSim’s array functions to reference a single item from the array. For example, to reference the value at 4 days for the 3rd item of a Time Series vector named X, you would use the following expression: getitem(X(4 day), 3).

**Read more:** Array Functions (page 801).

In order to gain a better appreciation for how to use Time Series elements, the manner in which the outputs are computed for each of the four kinds of input data is described in the sections below, along with an example for each kind of input. The specific examples mentioned here can be viewed in the file named BasicTimeSeries.gsm, which can be found in the TimeSeries subfolder of the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

**Time Series Examples**

**Example: Time Series Inputs Represent Instantaneous Values**

This example illustrates the use of a Time Series where the input data represent the instantaneous value of a variable at specified times. This is useful for simulating such things as the price of a commodity (e.g., gold), the price of an asset (e.g., a stock), or the height of a projectile.

The example can be viewed in the file named BasicTimeSeries.gsm, which can be found in the TimeSeries subfolder of the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

In this example, the input data represent the instantaneous price of a commodity (e.g., gold) at specified points in time:
The actual data set looks like this:

![Data Set](image)

The Primary Output is computed by linearly interpolating between the specified points at every update (i.e., timestep).

For example, at Time = 250 days, the Value output would be $274.50 (halfway between 279 and 270). The Rate_of_Change output is computed by differentiating the specified values. Hence, between 200 and 300 days, the Rate is -$9/100 days = -0.09 $/day.

The Primary Output and Rate_of_Change outputs for this particular example are shown below:
Using Time Series Elements

• Chapter 4: Using the GoldSim Elements

GoldSim User's Guide

**Note:** If the data are specified to represent an "Instantaneous value", the last data point must be at or beyond the end of the simulation. That is, if you have specified time entries as elapsed times, the last data point must be greater than the simulation duration. If you have specified time entries as dates, the last date must be equal to or after the end time for the simulation.

**Example: Time Series Inputs Represent Constant Values Over Specified Intervals**

This example illustrates the use of a Time Series where the input data represent constant values over specified intervals. That is, the data "stair-steps" from interval to interval. In some instances, this may reflect the fact that the variable actually is held constant over each interval. In other cases, it may be an artifact of how the data was recorded (e.g., it may have only been recorded in terms of an average value over the interval).

The example can be viewed in the file named BasicTimeSeries.gms, which can be found in the TimeSeries subfolder of the General Examples folder in your GoldSim directory (accessed by selecting **File | Open Example...** from the main menu).

In this example, the input data represent the average temperature over specified time periods:
The actual data set looks like this:

<table>
<thead>
<tr>
<th>Elapsed Time [day]</th>
<th>Value [°C]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7.0</td>
</tr>
<tr>
<td>2</td>
<td>7.0</td>
</tr>
<tr>
<td>3</td>
<td>10.2</td>
</tr>
<tr>
<td>4</td>
<td>12.1</td>
</tr>
<tr>
<td>5</td>
<td>13.0</td>
</tr>
<tr>
<td>6</td>
<td>15.3</td>
</tr>
<tr>
<td>7</td>
<td>18.5</td>
</tr>
<tr>
<td>8</td>
<td>17.9</td>
</tr>
<tr>
<td>9</td>
<td>18.2</td>
</tr>
<tr>
<td>10</td>
<td>13.7</td>
</tr>
<tr>
<td>11</td>
<td>10.6</td>
</tr>
<tr>
<td>12</td>
<td>9.3</td>
</tr>
<tr>
<td>13</td>
<td>9.3</td>
</tr>
<tr>
<td>14</td>
<td>9.9</td>
</tr>
<tr>
<td>15</td>
<td>9.9</td>
</tr>
</tbody>
</table>

The Primary Output is computed by assuming that the specified values stay constant over the time intervals between data points. That is, GoldSim does not interpolate between the data points. Hence, in this example, between 59 and 90 days, the temperature is assumed to be constant at 10.2 °C.
**Note:** There are two options for representing time series that represent constant values over specified intervals. You can specify that the data is constant over the *next* interval (as specified above), or you can specify that the data is constant over the *previous* interval. In the latter case, the data set shown above would imply that between 59 and 90 days, the temperature is assumed to be constant at 12.1°C. (In order for the two representations to provide the same results, you would need to shift the data column by one row.)

The Primary Output and the Rate_of Change outputs for this particular example are shown below:

![Temperature Plot](image.png)

In this example, the Rate_of Change output is also computed, although for this type of input data (constant values over intervals), this would typically not be done. The Rate_of Change is computed by differentiating the Values. Because the value is assumed to remain constant between data points, and then jumps discontinuously, the Rate is zero between data points, and then “spikes” when the Value changes. Theoretically, the derivative at this point is infinite (the curve is a vertical line). GoldSim computes the derivative as the difference between values divided by the timestep length. As such, the Rate_of Change is somewhat arbitrary. Note, however, that due to the way the Rate_of Change is computed, if you were to integrate this, it would correctly reproduce the Primary Output.
Note: If the data is specified to represent a "Constant value over the next interval", the first data point must be at or before the beginning of the simulation. That is, if you have specified time entries as elapsed times, the first data point must be less than or equal to zero. If you have specified time entries as dates, the first date must be at or before the start time for the simulation. Similarly, if the data is specified to represent a "Constant value over the previous interval", the last data point must be at or beyond the end of the simulation. That is, if you have specified time entries as elapsed times, the last data point must be greater than the simulation duration. If you have specified time entries as dates, the last date must be at or after the end time for the simulation.

Example: Time Series Inputs Represent Changes Over Specified Intervals

This example illustrates the use of a Time Series where the input data represent changes over specified intervals. This option is usually used to output cumulative values (e.g., cumulative rainfall, cumulative sales) and/or rates (e.g., sales rates, rainfall rates) that are specified by defining incremental changes in a variable (e.g., monthly sales, monthly rainfall) over specified intervals.

The example can be viewed in the file named BasicTimeSeries.gsm, which can be found in the TimeSeries subfolder of the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

It is important to understand that when this option is selected, the Primary Output of the Time Series is not the specified time series itself. Rather, it is the cumulative value of the variable at each point in the simulation (assuming that the specified changes occur uniformly over each interval). This option therefore typically takes advantage of the Rate_of_Change output.

The Rate_of_Change output for a Time Series that is defined as a series of changes over specified intervals represents the rate of change of the cumulative value. For example, if the data was entered as monthly sales increments, the Rate_of_Change would represent the sales rate.

The rate is computed by dividing the incremental change in the variable over the interval by the interval length. GoldSim assumes that the variable (e.g., cumulative rainfall or sales) changes uniformly over the time period, which means that the rate of change is constant over the interval (i.e., it "stair-steps").

Note: This option is particularly useful for time series that are collected at variable time points (e.g., once per month, since the number of days in a month is not constant). GoldSim is able to readily turn such data into rates.

In this example, the input data represent the total rainfall over specified time periods:
The actual data set looks like this:

![Time Series Data Table]

In this example, between 59 and 90 days, the value is assumed to have changed by 3.5 mm.

**Note:** There are two options for representing time series that represent changes over specified intervals. You can specify that the data changes over the next interval (as specified above), or you can specify that the data changes over the previous interval. In the latter case, the data set shown above would imply that between 59 and 90 days, the value changed by 3.7 mm. (In order for the two representations to provide the same results, you would need to shift the data column by one row.)
The Primary Output and the Rate_of_Change outputs for this particular example are shown below:

![Changes over an Interval (Monthly Rainfall)](image)

The Primary Output is computed by integrating the changes, assuming that the changes occur uniformly over each interval. Note that this implies that the rate of change is constant over the interval (i.e., it "stair-steps"). This is the appropriate way to represent variables that are measured on a monthly basis (e.g., sales, rainfall) and need to be converted into rates for use in your model (e.g., sales rate, rainfall rate).

**Note:** If the data is specified to represent a "Change over an interval" (either next or previous), then the first data point must be at or before the beginning of the simulation, and the last data point must be at or beyond the end of the simulation. That is, if you have specified time entries as elapsed times, the first data point must be less than or equal to zero, and the last data point must be greater than or equal to the simulation duration. If you have specified time entries as dates, the first date must be at or before the start time for the simulation, and the last date must be at or after the end time for the simulation.

**Example: Time Series Inputs Represent Discrete Changes**

This example illustrates the use of a Time Series where the input data represent discrete changes at specified times. It can be viewed in the file named BasicTimeSeries.gsm, which can be found in the TimeSeries subfolder of the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

In this example, the input data represent discrete changes to a price (in this case, gasoline):
Using Time Series Elements

When the input data represent discrete changes to a variable at specified times, the Primary Output is a discrete change signal (with an Add Instruction). Only certain elements in GoldSim can accept discrete change signals as inputs.

Read more: Propagating Discrete Signals Between Elements (page 353); Generating Discrete Changes Using Time Series Elements (page 415).

Discrete change signals are instantaneous quantities that cannot be directly plotted. They are used to instantaneously change the value of a quantity (represented by an Integrator, a Reservoir or a Pool element). In this example, an Integrator is used to represent the current gasoline price. It is modified by the output of the Time Series:
Read more: Integrator Elements (page 228).

The output of the Integrator looks like this:

![Integrator Properties: Gasoline, Price](image)

**Note:** Unlike other kinds of Time Series, when the input data represent discrete changes, it is not necessary for the first data point to start prior to or at the start of the simulation.

**Advanced Time Series Options**

GoldSim provides five advanced options for defining and using Time Series elements:

- You can enter historic time series data and then instruct GoldSim to adjust the time/date column (time shift) in a specified way for each simulation (e.g., in order to start at a random point in the data series, or
to shift the series forward so that historic dates are appropriately mapped onto the simulated dates).

- You can specify multiple sets of time series data within a single Time Series element, and then specify which series is to be used for any particular simulation. For example, this allows different time series data to be used for each separate realization in a Monte Carlo simulation.

- You can read the output of any other element in GoldSim, and "record" the results, and then "play them back" in a subsequent run of the model. Among other applications, this can be used to copy results from one model and use them as inputs for another.

- You can link to a "time series definition" that was output by another Time Series element within an embedded SubModel or from an external function (a DLL). The primary application of this advanced option is to transfer time history data between SubModels.

These features are discussed in detail in the sections below.

Many of these advanced features are accessed by pressing the More button on the Time Series dialog, which provides access to the “Advanced” section of the dialog:

**Time Shifting Time Series Data**

In many cases, you will want to use historic data to populate a Time Series. Having done so, you then want to apply that historic data to your simulation (which likely looks forward in time). In order to do this, therefore, you need to time shift the data so that the historic data is applied in an appropriate and consistent manner.

To support this, GoldSim provides an option to time shift a data series in an appropriate manner. This feature is provided in the “Advanced” section of the Time Series dialog, accessed via the More button:

If you check **Enable Time Shifting of Time Series Data**, the Settings… button becomes available, providing access to this dialog (in fact, when you first check the Time Shifting box, this dialog opens automatically):
Note: Time shifting is not available if the input represents a “Change over the next time interval”, a “Change over the previous time interval” or a “Discrete Change”.

GoldSim provides two different ways that Time Series can be shifted.

The first option is to **Use random starting point**. This option randomly samples a starting point in the data set for each realization. This is useful, for example, if you have 50 years of rainfall data, you want to carry out a 1 year simulation, and you want GoldSim to randomly sample a different historic year for each realization.

The manner in which GoldSim randomly samples the starting point in a data set is controlled by two things:

- The **Data periodicity**, which is either “none”, “annual”, or “diurnal”; and

- Whether or not the time series data is specified in terms of Dates (Calendar Time) or Elapsed Time.

**Read more:** [Entering Time Series Data as Dates or Elapsed Times](page 197).

The behavior is summarized below:

- If the **Data periodicity** is “none”, for each realization GoldSim randomly picks a point from within the total span of the time series data, and assumes that this time point corresponds to an elapsed time of zero. Note that this means that if data is entered as Calendar Time, the date-time is not respected; GoldSim randomly selects a point in time (and corresponding data value) from the time series data set, and “shifts” this point in time so that in the simulation it becomes an elapsed time of zero (i.e., the Simulation Start Date).

- If the **Data periodicity** is “annual”, GoldSim ensures that the random starting point for each realization is sampled such that all starting points are a multiple of 1 (average) year apart. In particular, random starting points for successive realizations are always multiples of 365.25 days apart (rounded to the nearest day). Hence,
  - If data is entered as elapsed time, the random starting point is either 0d, 365d, 731d, 1096d, etc.).
  - If data is entered as dates (Calendar Time), the random starting points for successive realizations would be generated by first
finding a data point in the data set that respected the month, day and time of the Simulation Start time, and then randomly shifting from that point by a multiple of 365.25 days (rounded to the nearest day). (Note that this implies that the shifted calendar day could be off by one day from the actual date).

**Note:** When specifying time shifting using annual periodicity, your time series should contain at least 366 days of data.

- If the **Data periodicity** is “diurnal”, GoldSim ensures that the random starting point for each realization is sampled such that all starting points are a multiple of 1 day apart. In particular,
  - If data is entered as elapsed time, random starting points for successive realizations are always multiples of 1 day apart. That is, the random starting point is either 0d, 1d, 2d, 3d, etc.).
  - If data is entered as dates (Calendar Time), random starting points for successive realizations respect the hour, minute and second of the Simulation Start time.

**Note:** The random sampling also takes into account the amount of data. For example, if you had 1000 days of data and selected “annual” Data periodicity, the probability of picking the first two years for the starting point would be (365/1000), while the probability of picking the last year would be only (270/1000). In this example, in order to have equal probability of picking any year as a starting point, you would require three full years of data.

The second option for time shifting Time Series is to **Shift data years to align with simulation dates**. This option is only available if the Time Series data is specified using Calendar Time.

This option shifts the time series data forward (or backward) by a multiple of a year such that the simulation begins by using data from the specified **Data year to start in**. For example, if the actual data set started on 31 July 1989 and ended on 31 December 2011, the Simulation Start Date was 1 January 2013, and the **Data year to start in** was entered as 1990, GoldSim would treat the data set such that the data point corresponding to 1 January 1990 would be used for 1 January 2013 (and, assuming daily data was entered, the data point for 2 January 1990 would be used for 2 January 2013, etc.).
This is useful, for example, if you wish to apply historic data to a forward-looking simulation, assuming no uncertainty or randomness. That is, this option assumes that with regard to the Time Series, the future will be exactly the same as the past (as represented by the historic data).

Note: If you wanted to account for uncertainty and randomness while still directly using historic data, you could do so by entering multiple years of data, and then using the Use random starting point option with “annual periodicity”.

In order to better understand how this works, the table below summarizes how the data would be shifted under various conditions. For this table, it is assumed that the actual time series data range is from 31 July 1989 to 31 December 2011, and the Simulation Start Date is 1 January 2013:

<table>
<thead>
<tr>
<th>Specified “Data year to start in”</th>
<th>Data point used to represent 1 January 2013</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1990</td>
<td>1 January 1990</td>
<td></td>
</tr>
<tr>
<td>1995</td>
<td>1 January 1995</td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td>Fatal error</td>
<td>There is no data point for 1 January 1989 (since the data set starts on 1 July 1989).</td>
</tr>
<tr>
<td>2012</td>
<td>Fatal error</td>
<td>There is no data point for 1 January 2012 (since the data set ends on 31 December 2011).</td>
</tr>
</tbody>
</table>

Several points should be noted regarding the use of this option:

- The Data year to start in must be specified such that the data set contains data (directly or via interpolation) in the specified year for the month and day defined by the Simulation Start Date. Otherwise, as shown in the table above, GoldSim will display a fatal error.
- The value entered for Data year to start in is always rounded to the nearest integer.
- Data year to start in can be specified as a function. One way this can be used is to specify a different year for every realization. For example, if Data year to start in was defined as “1990 + Realization - 1”, data would be used starting from 1990 for the first realization, 1991 for the second realization, and so on.
- When you use this option, you are implicitly stating that the data has annual periodicity. This has implications for how GoldSim wraps data (as pointed out in the second bullet below).

When you choose to Enable Time Shifting of Time Series Data, GoldSim automatically wraps around to the start of the data set if the end of the data set is reached during a simulation. The manner in which this is done depends on the periodicity of the data:

- If using a random starting point with no periodicity, when the end of the data set is reached, the data set is effectively replicated and shifted
Using Time Series Elements

Using Time Series Elements

foward by a period of time equal to the span of the data set. For example, if the last data point was at 1000 days, and the simulation extended beyond 1000 days, the data set would be shifted forward by 1000 days, so that the initial starting value (e.g., elapsed time = 0) would correspond to 1000 days.

- If using a random starting point with annual periodicity or if aligning data years with simulation dates, when the end of the data set is reached, the data set is effectively replicated and shifted forward by N years, where N is the number of whole years in the data set. For example, if there were 3.5 years of data in the data set, the last data point was on December 31, and the simulation extended beyond 3.5 years, the data set would be shifted forward by 3 years, so that the first occurrence of January 1 in the data set would follow the last point in the data set.

- If using a random starting point with diurnal periodicity, when the end of the data set is reached, the data set is effectively replicated and shifted forward by N days, where N is the number of whole days in the data set. For example, if there were 21.5 days of data in the data set, the last data point was on day 21.5 days, and the simulation extended beyond 21.5 days, the data set would be shifted forward by 21 days, so that the data point at 0.5 days would follow the last point in the data set.

The example file (TimeShifting.gsm) in the TimeSeries subfolder of the General Examples folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) includes examples on the use of time shifting Time Series elements.

One of the advanced features of Time Series elements is the ability to specify multiple sets of time series data within a single Time Series element, and then specify which series is to be used for any particular simulation. This provides a way, for example, for different time series data to be used for each separate realization in a Monte Carlo simulation.

This feature is provided in the “Advanced” section of the Time Series dialog, accessed via the More button:

Defining Multiple Time Series in a Single Time Series Element

If you check Enable Multiple Series, there are two changes to the dialogs:

1. The Active Series field becomes available, and defaults to “Realization” (this will be discussed further below); and

2. The time series editing dialog accessed via the Edit Data button is modified to support specifying multiple series.

This expanded editing dialog (with two additional series added in this case) looks like this:
In addition to the two original buttons (add and remove rows), there are six new buttons at the top of the dialog:

<table>
<thead>
<tr>
<th>Button</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Add" /></td>
<td>Add new row to current series</td>
</tr>
<tr>
<td><img src="image" alt="Remove" /></td>
<td>Remove selected row from current series</td>
</tr>
<tr>
<td><img src="image" alt="Add" /></td>
<td>Add new series</td>
</tr>
<tr>
<td><img src="image" alt="Remove" /></td>
<td>Remove current series</td>
</tr>
<tr>
<td><img src="image" alt="Go" /></td>
<td>Go to previous series (hold Ctrl key to go to first series)</td>
</tr>
<tr>
<td><img src="image" alt="Go" /></td>
<td>Go to next series (hold Ctrl key to go to last series)</td>
</tr>
<tr>
<td><img src="image" alt="Go" /></td>
<td>Go to series at a specified index (dialog will prompt for series number)</td>
</tr>
<tr>
<td><img src="image" alt="Go" /></td>
<td>Change the current series number to a specified number (dialog will prompt for series number). If the specified series is higher than the current series, the specified series (and all series before it) are shifted down. If the specified series is lower than the current series, the specified series (and all series after it) are shifted up. (Instead of entering a number, you can also hold the Ctrl key and press the button to shift the data series to the next index; or also hold the Ctrl and Shift keys and press the button to shift the data series to the previous index.)</td>
</tr>
</tbody>
</table>

These buttons allow you to create, remove, edit and navigate the various time series data sets.

As you are moving between time series, the current series is indicated in the Title bar (e.g., “Series 2 of 3”).

The Description field can be valuable when creating multiple series, as it provides a mechanism for you to document what that particular series represents (e.g., “Scenario 3”).

The Time Unit that is specified in the dialog is always applied to all the series. That is, if you change it for one series, it is changed for all series.

**Note:** When you have multiple series, they do not have to have the same time points, or even the same duration or data span (although they often will). Of course, as is the case for any Time Series element, if the time series data does not contain appropriate starting and end points, it may trigger an error message at run time.
Once you have specified multiple time series, for any given simulation, you must specify which series to use via the **Active Series** field. This field must be a scalar dimensionless value that corresponds to one of the defined time series. Note that the time series are numbered sequentially, starting with 1. So, for example, if you had defined 10 series, then **Active Series** would need to be a number between 1 and 10, inclusive.

GoldSim will round off real numbers that are entered in the field (e.g., 2.3 would be interpreted as 2). However, if the rounded number does not correspond to one of the defined series numbers, GoldSim issues a fatal error.

The two most common ways that multiple time series are likely to be used are to 1) represent alternative input scenarios; and 2) to represent multiple realizations.

For example, if you had six different scenarios for a model (each with a different input time series), you could represent this by defining six series in a Time Series element, and setting the **Active Series** to “Scenario”, which would be an element in the model that took on values of 1 through 6 (i.e., a Scenario Data element).

**Read more:** [Scenario Data: Defining Inputs for Different Scenarios](page 507).

To use multiple time series to represent different realizations, you could define a different series for each Monte Carlo realization, and then set **Active Series** to “Realization”. In fact, this is the default setting when you activate multiple realizations:

Why would you want to use different time series for each realization?

There are two primary reasons why you might want to enter a large number of time series, each representing a different realization:

1. To represent uncertainty in time series inputs. For example, you could generate (using another program) 100 different random time series records. Every realization would then use a different time series.

**Note:** Currently, spreadsheets *cannot* be used to automatically import multiple time series. That is, if a Time Series is linked to a spreadsheet, it can only import a single series. Importing multiple time series from a spreadsheet into a Time Series element is not supported. As a result, if you needed to import a large number of separate time series records into a Time Series element, you would need to manually copy and paste the series into each element.

**Read more:** [Importing Data into a Time Series from a Spreadsheet](page 194).

2. To use the probabilistic results of one model as the input for a completely different model. For example, the first model might represent one part of the system whose only impact on the rest of the system (represented by a second model) was the time history output of
a single element in the first model. If there was no feedback from second model back to the first, the two models could be run separately in series. Importing the multiple time history realizations of the connecting element into a Time Series element in the second model provides the linkage between the two models.

To support this second application, GoldSim provides a mechanism to import time history realizations into a Time Series as multiple series. In particular, you can read the output of any other element in a GoldSim model, and “record” the results, and then “play them back” in a subsequent run of the model.

Read more: Using a Time Series Element to Record and Play Back a History (page 221).

This feature is described in detail in the section below.

The example file (MultipleSeries.gsm) in the TimeSeries subfolder of the General Examples folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) includes examples of Time Series elements with multiple series.

Time Series elements provide an advanced option in which you can read the output of another element in GoldSim, and “record” the results.

To access this capability, the Data Source should be set to ”Recording output history”. When you select this option, a new tab appears (Recording) from which you specify the link from the other element, as well as several additional options regarding how the output is recorded:

The Record from field represents the output that you wish to record. It must have the same dimensions and Type as specified on the main page (Definition tab) of the Time Series dialog.

The manner in which GoldSim records Time Series is controlled by how you have defined what the Time Series Represents, and whether or not the Trigger recording option is selected:
Using Time Series Elements

### Table: Time Series Represents, Trigger recording, Default Recording Logic Used by GoldSim

<table>
<thead>
<tr>
<th>Time Series Represents</th>
<th>Trigger recording</th>
<th>Default Recording Logic Used by GoldSim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instantaneous value</td>
<td>On</td>
<td>Records the value at start and end of simulation, and whenever triggered.</td>
</tr>
<tr>
<td>Instantaneous value</td>
<td>Off</td>
<td>Records the value at start and end of simulation, and at all scheduled timesteps if value has changed.</td>
</tr>
<tr>
<td>Constant value over the next (or previous) time interval</td>
<td>Not applicable (always Off)</td>
<td>Records the average value over the intervals at the start and end of simulation, and when triggered.</td>
</tr>
<tr>
<td>Change over the next time (or previous) interval</td>
<td>Not applicable (always On)</td>
<td>Records the change in the value over the intervals at the start and end of simulation, and when triggered.</td>
</tr>
<tr>
<td>Discrete change</td>
<td>Not applicable (always Off)</td>
<td>Records every discrete change.</td>
</tr>
</tbody>
</table>

**Read more:** [Specifying What the Input to a Time Series Represents](#) (page 190).

When **Trigger recording** is checked (either manually or by default based on what the Time Series represents), events that trigger a recording are specified by pressing the **Record** button:

![Trigger recording button](#)

This provides access to a standard Triggering dialog.

**Read more:** [Understanding Event Triggering](#) (page 355).

When recording Time Series, you should follow these general guidelines to ensure that the data is recorded in such a way that it fully captures the actual behavior of the output:

- If the output that is being recorded represents a quantity (e.g., the output of a Reservoir), the Time Series **Represents** should be set to "Instantaneous value".

- For most other outputs (and particularly for outputs that represent flows of material), the Time Series **Represents** should be set to "Constant value over the next (or previous) interval".

Note that the recording logic listed in the preceding table represents the minimum set of data points that GoldSim records. When the Time Series represents Instantaneous data, you can instruct GoldSim to record additional data by selecting one or more of the options below.

**Record at unscheduled timesteps (inserted by events).** When recording, the Time Series adds a time point at each scheduled timestep in the simulation. Some elements (that generate events) can also insert "unscheduled" timesteps (in between the scheduled timesteps). If this box is checked (the default), the Time Series records (adds a timepoint) at these unscheduled events also.

**Read more:** [Understanding Timestepping in GoldSim](#) (page 455).
**Record at all timesteps even if the input value is unchanged.** By default, when recording for some types of outputs (see above), the Time Series does not necessarily record every time point. If this box is checked, the Time Series records (adds a timepoint) at all timesteps.

The final two options only apply if the input data to the Time Series represent discrete changes. In this case, the **Record from** link specified must be a Discrete Change Signal. Note that multiple discrete change signals can be entered into this field by separating them with a semicolon.

**Record discrete changes with zero value.** This determines whether or not discrete changes with a zero value are recorded as time points. This box is defaulted on.

**Combine discrete changes that arrive simultaneously.** If two or more discrete changes that are being recorded by this Time Series occur at the same time, this checkbox determines whether the discrete changes are combined (the default), or if both are to be recorded separately.

While you are recording a Time Series, the element passes through as its output the instantaneous value of the input being recorded.

**Note:** Even if **Enable Rate of Change output** is checked, the Rate of Change output is not calculated while recording the history (it is output as zero).

**Note:** Even if the **Primary Output** is set to “Average value over the next timestep”, the output of a Time Series that is currently recording is always the instantaneous value.

Recording time series can also be used to record multiple series. To do so, you must check **Enable Multiple Series** on the Time Series dialog:

![Time Series Multiple Series Dialog](image)

Note that when enabling multiple series when recording, the **Active Series** field defaults to “Realization” and cannot be changed.

**Read more:** [Defining Multiple Time Series in a Single Time Series Element](page 218).

There are three primary applications for a “recording” Time Series:

- By recording the output of an element and then changing the **Data Source** field back to "Locally defined data", you can "play back" the history in a subsequent run of the model.
- You can record the times of "unscheduled" updates.
- You can pass the output of a Time Series that is "recording" to another Time Series element. The primary application of this is to transfer time histories between SubModels.

These applications are discussed briefly below.
Recording and Playing Back a History

This can be useful, for example, if you want to copy the result of one model into another model. In order to use a Time Series in this way, you would need to do the following:

1. Insert one or more Time Series elements (one for each output you are interested in), set their Data Source to "Recording output history", and specify the output that you wish to record on the Recording tab.

2. Run the model. After you run the model, copy and paste the Time Series element into a different model, and set Data Source to “Locally defined data”.

3. The Time Series elements you have pasted will contain the results from the other model and can now be used as input for the current model.

Recording the Time of Unscheduled Events

Because GoldSim only saves results at scheduled timesteps, the impact of "unscheduled" timesteps that are inserted by events can sometimes be difficult to see directly.

Read more: Understanding Timestepping in GoldSim (page 455).

A recording Time Series element, however, provides a way to actually see when these "unscheduled" timesteps occur. To use the Time Series in this way, you would need to do the following:

1. Insert a Time Series element into your model, set the Represents field to “Instantaneous value”, set the Data Source to "Recording output history", and specify an output. If you are only interested in the time of the "unscheduled" timesteps themselves, the output that you specify is not important (e.g., you could simply choose Etime).

2. Make sure that Record at unscheduled timesteps (inserted by events) is checked on.

3. After the simulation is over, by pressing the View Data button, you will be able to view a table of times for all of the scheduled and unscheduled timesteps.

Note: Although recording Time Series can be used in this manner, under most circumstances, there is a much easier way to achieve the same thing. GoldSim provides an option (under a specified set of conditions) to view the values of selected outputs at unscheduled updates in the Advanced Time settings.

Read more: Including Unscheduled Updates in Time History Results (page 477).

Transferring a Time History Out of a SubModel

In some situations, you may want to run a SubModel in GoldSim, and then pass the entire time history of one or more outputs to a location outside of the SubModel (e.g., to another SubModel).

Read more: Using SubModels to Embed Models Within Models (page 973).

To take advantage of this feature, you must use the Time History Definition output of the element. This is a complex output that contains the complete definition for the element and can be passed from the element (through a SubModel interface) to another Time Series element.
Read more: Browser View of a Time Series Element (page 227); Referencing a Time Series Definition Output (page 225).

To use the Time Series in this way, you would need to do the following:

1. Insert one or more Time Series elements into your SubModel, set their Data Source to "Recording output history", and specify the outputs that you wish to transfer outside of the SubModel.

2. Add the Time Series Definition Output of each Time Series that you wish to transfer to the output interface of the SubModel.

   Read more: Creating the Output Interface to a SubModel (page 984).

3. For each Time Series that you wish to transfer from the SubModel, you must define a corresponding "play back" Time Series element outside of the SubModel. For these Time Series, set their Data Source to "Linked to external Time Series Definition", and on the Linked tab, enter the Time Series Definition output from the SubModel interface.

4. The output of these Time Series elements will directly reflect the time histories generated by the Time Series elements inside the SubModel.

Note: Although recording Time Series can be used in this manner to transfer a time series out of a SubModel, under most circumstances, there is a more straightforward way to achieve the same thing. GoldSim provides an option to directly add an output inside a SubModel as a Time Series Definition to the SubModel interface. This allows you to skip step #1 above. The primary motivation to use a recording Time Series rather than adding the output directly to the SubModel interface would be if it was necessary to capture high resolution results (i.e., unscheduled updates) from inside the SubModel.

Read more: Including Unscheduled Updates in Time History Results (page 477).

The example file RecordingLinkingTimeSeries.gsm in the TimeSeries subfolder of the General Examples folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) includes examples of recording Time Series elements.

Time series elements provide an advanced option in which you can read a "Time Series Definition" output from another Time Series element or from an external function (a DLL).

To access this capability, the Data Source should be set to "Linked to external Time Series Definition". When you select this option, a new tab appears (Linked) from which you specify the link from the other element:
Using Time Series Elements

The **Linked to** field only accepts Time Series Definition outputs.

*Read more:* [Browser View of a Time Series Element](page 227).

This is a complex output that can only be produced by another Time Series element, by an external function, or by a SubModel.

---

**Note:** This field will not accept a Time Series Definition output directly from another Time Series. It will only accept Time Series Definition outputs from the output interface of a SubModel, or Time Series Definition outputs produced directly by an external function (a DLL).

---

*Read more:* [Using SubModels to Embed Models Within Models](page 973).

The primary application of this Time Series feature is to transfer time history data between SubModels. In this case, it is used in conjunction with a Time History Definition output on the output interface of a SubModel.

*Read more:* [Creating the Output Interface to a SubModel](page 984).

Occasionally, rather than reading the time history directly into a Time Series outside of the SubModel, you may wish to first process that data in some way. This is facilitated by allowing external functions (DLLs) to read and write Time Series Definitions.

*Read more:* [External (DLL) Elements](page 931).

---

**Note:** If using an External element to output a Time Series Definition, the DLL can only be called at etime = 0. That is, because a Time Series must be defined at the beginning of the simulation, if you try to redefine the time series in the middle of the simulation, GoldSim will issue a Fatal Error.

To use the Time Series in this way, you would need to do the following:
1. Insert one or more Time Series elements into your SubModel, set their **Data Source** to "Recording output history", and specify the outputs that you wish to transfer outside of the SubModel.

2. Add the Time Series Definition Output of each Time Series that you wish to transfer to the output interface of the SubModel.

   **Read more:** [Creating the Output Interface to a SubModel](#) (page 984).

3. For each Time Series that you wish to transfer from the SubModel, you must define a corresponding Time Series element outside of the SubModel. For these Time Series, set their **Data Source** to "Linked to external Time Series Definition", and on the **Linked** tab, enter the Time Series Definition output from the SubModel interface.

   **Note:** If you wanted to process the data in some way, after exporting it from the SubModel, you could read the Time Series Definition into a DLL that subsequently passed an updated Time Series Definition to another Time Series element.

4. The output of these Time Series elements will reflect the time histories generated by the Time Series elements inside the SubModel.

   The example file RecordingLinkingTimeSeries.gsm in the TimeSeries subfolder of the General Examples folder of your GoldSim directory (accessed by selecting **File | Open Example...** from the main menu) includes an examples of a Time Series element reading in a Time Series definition input from a SubModel.

   The browser view of a Time Series element shows at least two, and in some cases, three outputs (the number of outputs produced by the element being user-determined):

   **Browser View of a Time Series Element**

   ![Browser View of a Time Series Element](image)

   **Note:** Element outputs are only shown in the browser if you choose to **Show Element Subitems** (accessed via the browser context menu by right-clicking in the browser).

   **Read more:** [Using the Browser](#) (page 105).

   The first output is the Primary Output for the Time Series. If the user has selected to **Enable Rate of Change output**, a Rate_of_Change output is also available.

   The Time Series Definition output is also always present, and is a complex output that can be used to transfer time series data between SubModels (or External elements).

   **Read more:** [Referencing a Time Series Definition Output](#) (page 225).
Because the entries to the table defining the data records must be numbers, they are not shown in the browser (since they cannot be linked to). Hence, within the browser, a Time Series element has no inputs.

**Using Stock Elements**

Stock Elements are perhaps the most important elements in GoldSim. Stock elements are important because they have the property that they accumulate past events and provide systems with inertia and memory, and hence are responsible for internally generating the dynamic behavior of a system. Without such elements, your models could respond to outside (exogenous) drivers, but could not generate any dynamics of their own. While there are other elements that behave in this way that are not Stocks (i.e., Delay elements), most (if not all) real-world dynamic systems will involve at least one Stock.

Stock elements have the special property that their outputs are influenced by what has happened in the past. That is, their outputs are determined by the previous values of their inputs.

GoldSim provides three types of Stock elements, listed below in terms of increasing complexity:

- Integrators are intended to integrate information. For example, integrating a time series of the velocity of a projectile using an Integrator would yield the height of the projectile.
- Reservoirs are intended to integrate/accumulate material flows and represent things such as reservoirs of water, financial account balances, and inventories.
- Pools are more powerful versions of Reservoirs (i.e., they are also intended to integrate/accumulate material flows). They have additional features to more easily accommodate multiple inflows and outflows.

**Integrator Elements**

Integrator elements are elements that integrate rates. You use them to integrate and track information, such the distance traveled by a car, interest rates, commodity prices, temperature, rainfall totals, consumer confidence, or the condition (reliability, strength) of a machine or a material. You can also use an Integrator element to calculate moving averages of the input signal (the Rate of Change).

Their default symbol is an integration sign because, mathematically, they represent integrals. An Integrator requires an **Initial Value** and a **Rate of Change**, and computes a single output (its Value) as follows:

\[
\text{Value} = \text{Initial Value} + \int (\text{Rate of Change}) \, dt
\]

The Rate of Change, of course, can be a function of time.

The properties dialog for an Integrator looks like this:
An example model which illustrates the use of Integrator elements (Integrator.gsm) can be found in the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

Numerically, GoldSim approximates the integral represented by an Integrator as a sum:

\[ \text{Value}(t_n) = \text{Initial Value} + \sum_{i=1}^{n} \text{Rate of Change}(t_i - \Delta t_i) \Delta t_i \]

where \( \Delta t_i \) is the timestep length just prior to time \( t_i \) (typically this will be constant in the simulation), and \( \text{Value}(t_n) \) is the value at the end of timestep \( n \). Note that the Value at a given time is a function of the Rate of Change at previous timesteps (but is not a function of the Rate of Change at the current time).

In order to better understand how GoldSim carries out this numerical integration, it is worthwhile to consider the following example. Consider a car that drives on a road for five minutes. We input the time history of the velocity of the car as follows:

![Graph of velocity over time](image)
Hence, it is stopped for the first minute, travels at 20 km/hr for the second minute, 60 km/hr for the third minute, 20 km/hr for the fourth minute, and 60 km/hr for the fifth minute.

If we put this time history into an Integrator element, and run the simulation for 5 minutes with a constant 1 minute timestep, we get the following for the cumulative distance traveled (assuming an Initial Value of 0):

(This example model, named Integrator.gsm, can be found in the General Examples folder in your GoldSim directory, accessed by selecting File | Open Example... from the main menu.)

The manner in which GoldSim computed these results is summarized in the following table:

<table>
<thead>
<tr>
<th>i timestep</th>
<th>ti (min)</th>
<th>Velocity During Next Step</th>
<th>Distance Traveled During Previous Step</th>
<th>Cumulative Distance Traveled at time = ti</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>0</td>
<td>0 km/hr</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>20 km/hr</td>
<td>0 km/hr * 1 min = 0 km</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>60 km/hr</td>
<td>20 km/hr * 1 min = 0.333 km</td>
<td>0.333 km</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>20 km/hr</td>
<td>60 km/hr * 1 min = 1 km</td>
<td>1.333 km</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>60 km/hr</td>
<td>20 km/hr * 1 min = 0.333 km</td>
<td>1.666 km</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>-</td>
<td>60 km/hr * 1 min = 1 km</td>
<td>2.666 km</td>
</tr>
</tbody>
</table>

Note that the Value (cumulative distance traveled) at time ti is independent of the Rate of Change (velocity) at time ti. It is only a function of the Rates (velocities) at previous timesteps.

This particular numerical integration method is referred to as Euler integration, and is discussed further in Appendix F. For the present purposes, it is sufficient to note that Euler integration assumes that the Rate of Change is constant over a
timestep. Hence, if you specify that the Rate of Change to an Integrator varies linearly from 0 km/hr to 10 km/hr over 10 minutes, and simulate the system with a 1 minute timestep, GoldSim will approximate the velocity curve internally as a stair-step function:

![Velocity Curve](image)

The dashed line represents the specified Rate of Change. The solid line illustrates how the Integrator element would interpret the Rate of Change, assuming a 1 minute timestep.

**Note:** By default, the Rate of Change at time t is treated as the constant rate of change over the next timestep (i.e., from t to t + Δt). GoldSim provides an advanced option to allow you to modify this behavior such that the Rate of Change at time t is treated as the constant rate of change over the previous timestep (i.e., from t to t - Δt).

**Note:** Euler integration is the simplest and most common method for numerical integration. However, if the timestep is large, this method can lead to significant errors. This is particularly important for certain kinds of systems in which the error is cumulative (e.g., sustained oscillators such as a pendulum). In these cases, these errors can be reduced by using Containers with Internal Clocks, which allow you to locally use a much smaller timestep.

**Read more:** Modifying How Integrators Treat the Rate of Change (page 232); Specifying Containers with Internal Clocks (page 474).

**Specifying the Inputs to an Integrator**

The Display Units determine the dimensions of the Integrator’s output. This output can only be specified as a value (it cannot be a condition), and can be specified as a scalar, a vector or a matrix. You can specify whether the Integrator is a scalar, vector or matrix by pressing the Type… button. By default, the output of a new Integrator element is a scalar, dimensionless value.

**Read more:** Using Vectors and Matrices (page 782).

The Initial Value input to the Integrator must have the same attributes (order and dimensions) as the output.
**Specifying Discrete Changes to an Integrator**

Integrators can also accept discrete (as opposed to continuous) changes. That is, Integrators actually compute their Current Value by accounting for discrete changes as follows:

\[
\text{Value}(t_n) = \text{Initial Value} + \sum_{i=1}^{n} \text{Rate of Change}(t_i - \Delta t_i)\Delta t_i + \sum \text{Discrete Changes}
\]

Discrete changes are specified by linking a *discrete change signal* input into the Discrete Change field.

*Read more:* [Basic Concepts of Discrete Event Modeling](#) (page 352); [Modeling Discrete Changes to an Integrator](#) (page 385).

**Modifying How Integrators Treat the Rate of Change**

By default, the Rate of Change at time \( t \) is treated as the constant rate of change over the *next* timestep (i.e., from \( t \) to \( t + \Delta t \)). In some very special cases, however, you may actually want GoldSim to treat the Rate of Change in a different manner. In particular, you may want to modify this behavior such that the Rate of Change at time \( t \) is treated as the constant rate of change over the *previous* timestep (i.e., from \( t \) to \( t - \Delta t \)).

The most common reason for doing this is when you want to use an Integrator to integrate a mass flux coming from a pathway element in the GoldSim Contaminant Transport Module. Pathway elements output mass flux rates that represent rates over the *previous* timestep. Therefore, in order to correctly integrate such a flux, an Integrator element would need to treat the Rate of Change as if it represented the rate of change over the previous timestep.
GoldSim provides an option to allow Integrators to treat the **Rate of Change** in this way. By default, the option is deactivated. To activate it, you must press the **More** button to expand the dialog:

By default, the checkbox labeled **Rate of change applies to the PREVIOUS time step** is cleared. If you check this checkbox, the Integrator element will treat the **Rate of Change** as if it represented the rate of change over the *previous* timestep.

In some cases you may want to represent a moving average in your model. For example, at any given time, you might want to know what the average value was over the previous 30 days.

You can use an Integrator to compute a moving average of the **Rate of Change** input.

A moving average is a uniformly-weighted average of the **Rate of Change** over a specified averaging time (ATime). It is calculated as the change in the value of the Integrator over the averaging time divided by the averaging time:

\[
\text{Moving Average} = \frac{\int_0^{\text{ETime}} \text{(Rate of Change)} \, dt}{\text{ETime} - \text{ATime}}
\]
**Note:** To account for elapsed times that are less than the averaging time, the actual averaging time used is the minimum of the elapsed time and the specified averaging time. The value of the moving average at elapsed time \( t = 0 \) is equal to the initial value of the Rate of Change.

GoldSim provides an option to compute up to three different moving averages. To activate it, you must press the More button to expand the dialog:

![Advanced Options](image)

By checking a box, you can specify the name of the moving average output and define the Averaging Period. If you specify multiple moving average outputs, they must have unique names. The Output IDs follow the same naming conventions as element names. The Averaging Period must have units of time, and must be a number or a link from a static variable (e.g., a constant Data element or some other output that does not change with time).

Discrete changes (applied in the Discrete Change field) will affect the moving average outputs. In particular, any discrete changes that are applied to the Integrator will affect the moving average as if they were distributed uniformly over the preceding timestep (i.e., from the previous update of the element to the update where the discrete change was applied).

**Note:** The moving average outputs have the same dimensions as the Rate of Change input (not the dimensions defined by the Display Units for the Integrator itself). For example, if the variable you are interested in computing a moving average for has dimensions of volume/time, then the Integrator would need to have dimensions of volume. Similarly, if the variable you are interested in computing a moving average for has dimensions of mass, then the Integrator would need to have dimensions of mass-time.

**Note:** The Display Units for the moving average outputs are the Display Units for the element divided by the Time Display Units (specified in the Simulation Settings dialog). If this results in unusual Display Units, you can simply route the output into an Expression with adjusted Display Units.

**Read more:** [Simulation Settings](#) (page 452).

**Using an Integrator to Model an Aging Chain**

In some situations, it is necessary to keep track of the age structure of a stock of material or objects. For example, you may want to track the number of people in each of a number of age groups, the number of people in a company at different experience levels (e.g., new hire, experienced, expert), or the number of trucks of different age groups on the road.
To model such a situation, you cannot use a single Stock (e.g., an Integrator). Rather, you must disaggregate the total stock into multiple categories (referred to as cohorts). Each cohort “graduates” to the next cohort over time (and can only move in one direction). However, each cohort may grow and shrink for other reasons (e.g., if you were modeling a population of people, a particular category could grow due to immigration, and shrink due to emigration and death). Note that these rates of growing and shrinking are likely a function of the specific cohort (e.g., death rates).

GoldSim provides several methods for modeling aging chains. One of these involves using an Integrator and processing a specialized discrete change signal (called a Push). The advanced option to Provide a Pushed_Out discrete change output is used for this method:

By default, the browser view of an Integrator element shows three inputs and a single output:

Several Advanced Options can add inputs and outputs. If the moving average options is used, the browser would display additional inputs and outputs (one for each moving average specified). If the option to provide a “Pushed_Out” output is selected, the browser would display an additional Discrete Change output.

**Note:** Element inputs and outputs are only shown in the browser if you choose to Show Element Subitems (accessed via the browser context menu by right-clicking in the browser).

**Reservoir Elements**

Reservoir elements are elements that accumulate flows. You use them to accumulate and track material, such as the quantity of water in a lake, the amount of soil in a stockpile, the number of people in a city, and the number of widgets in a warehouse.

Their default symbol is a container of water since this is an excellent analogy for the behavior of the element. Like a container of water, a Reservoir element has a current value (e.g., the volume of water in the container), inflows, withdrawals and overflows.
A Reservoir is fundamentally similar to an Integrator (since both are Stocks), but is different in three important ways:

- Reservoirs are designed to integrate (accumulate) material; Integrators are designed to integrate information.
- Reservoirs explicitly differentiate between positive rates and discrete changes (additions) and negative rates and discrete changes (withdrawals).
- Reservoirs allow you to specify upper and lower bounds. Due to these bounds, a Reservoir has more than one output.

**Note:** If you are modeling a system that has multiple distinct withdrawals (particularly if the Reservoir can reach its Lower Bound), you likely will want to use a more powerful version of the Reservoir, referred to as a Pool.

**Read more:** Pool Elements (page 252).

Like an Integrator, a Reservoir requires an Initial Value and a Rate of Change. The Rate of Change, however, is specified in terms of two separate inputs, one representing the Rate of Addition and one representing the Rate of Withdrawal.

In the absence of Upper and Lower Bounds, the primary output of the Reservoir (its Value) is computed as follows:

$$ Value = Initial \ Value + \int (Rate \ of \ Addition - Rate \ of \ Withdrawal) \ dt $$

The Rates of Addition and Withdrawal can, of course, be functions of time.

The Properties dialog for a Reservoir element looks like this:
By default, when you create a new Reservoir element, a **Lower Bound** of 0 is assumed. You can subsequently remove the **Lower Bound**, or modify it.

An example model which illustrates the use of Reservoir elements (Reservoir.gsm) can be found in the General Examples folder in your GoldSim directory (accessed by selecting **File | Open Example...** from the main menu).

Numerically, GoldSim approximates the integral represented by a Reservoir as a sum:

\[
\text{Value}(t_n) = \text{Initial Value} + \sum_{i=1}^{n} \text{Rate of Change}(t_i - \Delta t_i) \Delta t_i
\]

where

\[
\text{Rate of Change}(t_i - \Delta t_i) = \text{Rate of Addition}(t_i - \Delta t_i) - \text{Rate of Withdrawal}(t_i - \Delta t_i)
\]

In these equations, \(\Delta t_i\) is the timestep length just prior to time \(t_i\) (typically this will be constant in the simulation), and \(\text{Value}(t_n)\) is the value at end of timestep \(n\). Note that the Value at time \(t_i\) is a function of the Rates of Addition and Withdrawal at previous timesteps (but is not a function of these Rates at time \(t_i\)).

Note that if you specify **Upper** or **Lower Bounds** for a Reservoir, the equation shown above is constrained by the specified bounds (i.e., the Value cannot exceed the **Upper Bound** and can not be less than the **Lower Bound**).

**Read more:** Defining Upper and Lower Bounds for a Reservoir (page 240).

Reservoirs and Integrator elements use the same numerical integration method, referred to as Euler Integration. A simple example illustrating this integration method is provided in the section discussing how an Integrator element computes its output.

**Read more:** How an Integrator Element Computes its Output (page 229).

The key assumption involved in this integration method is that at any point in time, the **Rates of Change** (Additions and Withdrawals) represent the rates over the next timestep, and those rates are assumed to be constant over the timestep. Euler Integration is discussed in additional detail in Appendix F.

**Note:** Euler integration is the simplest and most common method for numerical integration. However, if the timestep is large, this method can lead to significant errors. This is particularly important for certain kinds of systems in which the error is cumulative (e.g., sustained oscillators such as a pendulum). In these cases, these errors can be reduced by using Containers with Internal Clocks, which allow you to locally use a much smaller timestep.

**Read more:** Specifying Containers with Internal Clocks (page 474).

The primary output of a Reservoir is its Value. The **Display Units** determine the dimensions of this output. The primary output of a Reservoir is always a value (it cannot be a condition), but can be specified as a scalar, a vector or a matrix. You can specify the order by pressing the **Type...** button. By default, the primary output of a new Reservoir element is a scalar, dimensionless value.

**Read more:** Using Vectors and Matrices (page 782).
**Note:** GoldSim encourages (but does not require) you to specify **Display Units** for a Reservoir whose dimensions are consistent with materials. In particular, it expects the units to have dimensions of mass, volume, energy, amount, currency, or to be dimensionless. When you first specify your **Display Units**, GoldSim will warn you if your units do not have one of these dimensions. GoldSim will not, however, prevent you from using other units.

The **Initial Value** input to the Reservoir must have the same attributes (order and dimensions) as the primary output.

**Note:** The **Initial Value** must be a number or a link from a static variable (e.g., a constant Data element or a Stochastic).

There are two **Rate of Change** inputs: one representing **Additions** and one representing **Withdrawal Requests**. These inputs must have the same order (i.e., scalar, vector, matrix) as defined for the element. Moreover, the dimensions must represent a rate of change of the primary output (e.g., if the primary output has units of mass, the **Rate of Change** inputs must have units of mass per time).

The **Rate of Change** inputs must be entered as non-negative values. Negative values for these inputs during a simulation will result in a fatal error.

**Note:** The specified **Rate of Change** inputs represent constant rates over the next timestep. Hence, if an **Rate of Change** was defined as “if(time > 10 day, 2 m³/day, 1 m³/day)”, and you were using a 1 day timestep, the rate would not actually change to 2 m³/day until time = 11 days. That is, since at time = 10 days, the if statement indicates that the rate is equal to 1 m³/day, GoldSim would assume that the rate was equal to 1 m³/day between 10 days and 11 days. If you wanted the rate to change at 10 days, you would write the if statement as “if(time >= 10 day, 2 m³/day, 1 m³/day)”.

**Note:** If the output of a Reservoir is specified as being an array (a vector or a matrix), the **Initial Value** and **Rate of Change** inputs must, by definition, themselves be arrays. Since you often may wish to enter a zero array (a vector or matrix of zeros) for one or more of these inputs, GoldSim allows you to leave any of these input fields blank, in which case it interprets the field as a zero array.

**Using the Withdrawal Rate Output of a Reservoir**

By default, a Reservoir element has two outputs: the Value (which has the same name as the element) and a Withdrawal_Rate. In the absence of a **Lower Bound**, the Withdrawal_Rate output is exactly equal to the **Withdrawal Requests/Rate of Change** input.

To illustrate the use of the Withdrawal_Rate output, consider an example in which Reservoir1 flows into Reservoir2 (i.e., the Withdrawal_Rate output from Reservoir1 is the **Additions/Rate of Change** input for Reservoir2). One way to build this model is as follows:
By using the Withdrawal_Rate output, however, this model can become more transparent:

As can be seen, by linking the Withdrawal_Rate output from Reservoir1 to the Additions/Rate of Change input of Reservoir2, it becomes visually much clearer that Reservoir1 flows into Reservoir2.

Although the visual advantage of using the Withdrawal_Rate output is important, there is a much more important reason to use this output. In the first example above, the Data element serves as both an input to Reservoir1 (as the Withdrawal Requests/Rate of Change) and an input to Reservoir2 (as the Additions/Rate of Change). Note, however, that as will be discussed in detail in the next section, if a Lower Bound is specified for the Reservoir, the Withdrawal_Rate output is not necessarily identical to the Withdrawal Requests/Rate of Change input. In particular, if Reservoir1 hits its Lower Bound, the Data element no longer necessarily represents the actual amount being added to Reservoir2. The actual amount being added might be restricted and hence might be less than Data element Outflow_from_1. This is because the Data element represents the requested rate of withdrawal, but this is not necessarily the actual rate of withdrawal that is possible (there might not be enough material available). As a result, such a structure could cause you to artificially create material in this model!

Read more: How a Reservoir Computes the Withdrawal Rate (page 245).
Note: Because ignoring the Withdrawal_Rate output when a Reservoir reaches its Lower Bound (and the withdrawal is restricted) could result in a significant error in your model, GoldSim generates a warning message whenever a Reservoir reaches its Lower Bound such that the requested withdrawal cannot be delivered AND the Withdrawal_Rate output is not referenced by another element.

Note: If you are modeling a system that has multiple distinct withdrawals (particularly if the Reservoir can reach its Lower Bound), you likely will want to use a slightly more powerful version of the Reservoir, referred to as a Pool.

Read more: Pool Elements (page 252).

Defining Upper and Lower Bounds for a Reservoir

One of the most powerful features of a Reservoir is that you can specify a Lower and/or Upper Bound. If the Lower Bound and Upper Bound boxes are cleared for a Reservoir, the Value is not constrained and is unbounded (as is the case for an Integrator).

Checking one or both of these boxes, however, provides additional functionality and has important impacts on the outputs for the element. In particular:

- If the Lower Bound is checked, GoldSim enforces that limitation on the Value. As a result, the Withdrawal_Rate output is not necessarily equal to the Withdrawal Requests/Rate of Change input. In this case, the Withdrawal_Rate output represents the actual rate of withdrawal from the Reservoir, which may be less than the specified Withdrawal Requests/Rate of Change (which represents the demand or requested rate of withdrawal).

For example, if the Reservoir is at the Lower Bound, and the Additions/Rate of Change is zero, the Withdrawal_Rate output must be equal to zero, regardless of the Withdrawal Requests/Rate of Change (there is no material available to be withdrawn). Similarly, if the Reservoir is at the Lower Bound, and the Additions/Rate of Change is equal to 10, the Withdrawal_Rate output can be no greater than 10, regardless of the Withdrawal Requests/Rate of Change (if there is no storage, you cannot withdrawal more than is being supplied).

By default, when you create a new Reservoir element, the Lower Bound is checked (and a value of 0 is assumed).

Note: If the Lower Bound is constant, the Value in the Reservoir will never fall below the Lower Bound. However, if the Lower Bound is changing with time, under some circumstances, it is possible for the Value of the Reservoir to fall below the Lower Bound. The primary purpose of the Lower Bound is to accurately compute the Withdrawal_Rate. If the Lower Bound exceeds the amount in the Reservoir (because the Lower Bound has changed during a timestep), the Reservoir will not permit any withdrawals until it returns to the Lower Bound (via additions).
**Note:** If you are modeling a system that has multiple distinct withdrawals (particularly if the Reservoir can reach its **Lower Bound**), you likely will want to use a slightly more powerful version of the Reservoir, referred to as a Pool.

**Read more:**  [Pool Elements](#) (page 252).

- If the **Upper Bound** is checked, GoldSim enforces that limitation on the Value. GoldSim also adds two new (secondary) outputs: Overflow_Rate (a value) and Is_Full (a condition). If the Value is below the **Upper Bound**, the Overflow_Rate output is equal to zero and the Is_Full output is False. If the Value is at the **Upper Bound**, the Overflow_Rate output is equal to the difference between the specified Addition Rate and the specified Withdrawal Rate and the Is_Full output is True.

**Read more:**  [Using the Is_Full Output of a Reservoir](#) (page 250).

**Note:** If the **Upper Bound** is constant, the Value in the Reservoir will never exceed the **Upper Bound**. However, if the **Upper Bound** is changing with time, under some circumstances, it is possible for the Value of the Reservoir to exceed the **Upper Bound**. If the **Upper Bound** is exceeded (because it has changed during a timestep), GoldSim will overflow the excess uniformly over the next timestep.

Most real-world Reservoirs have such bounds. For example, many Reservoirs would have a logical lower bound of zero (e.g., a pond cannot contain a negative quantity of water, a parking lot cannot contain a negative number of cars). Many Reservoirs also have a logical upper bound (e.g., tanks and parking lots have maximum capacities). Some Reservoirs may have a lower bound, but no upper bound (e.g., a bank perhaps could not be negative, or at least there will be some limit on the amount that can be overdrawn, but would probably have no practical upper bound).

**Note:** If your specified **Upper Bound** is reached during a simulation, and the Overflow_Rate output is not referenced by any other element (e.g., as an inflow to another Reservoir), GoldSim will display a warning, as this could indicate that you have neglected to track flows leaving the Reservoir.

**Note:** GoldSim generates a warning message whenever a Reservoir reaches its **Lower Bound** such that the requested withdrawal cannot be delivered AND the Withdrawal_Rate output is not referenced by another element.

The Reservoir element is particularly powerful because it enforces the bounds while properly conserving the material being tracked. That is, if a Reservoir’s **Additions/Rate of Change** causes it to reach its **Upper Bound**, GoldSim produces an Overflow_Rate (which can be routed to a downstream Reservoir). Similarly, if you specify a **Withdrawal Requests/Rate of Change** that exceeds what can actually be delivered, GoldSim accurately reports the actual Withdrawal_Rate.
Furthermore, as is discussed below, the Upper and Lower Bounds can be specified as functions of time. This allows you to simulate systems such as a pond which is filling up with silt and sand (and hence decreasing its capacity), and a warehouse whose capacity is being expanded (via construction) throughout a simulation.

The Lower Bound and the Upper Bound inputs to the Reservoir must have the same attributes (order and dimensions) as the Reservoir’s primary output. In addition, the following limitations imposed on the inputs should be specifically noted:

- The Upper Bound must be greater than or equal to the Lower Bound.
- The Initial Value must be greater than or equal to the Lower Bound and less than or equal to the Upper Bound.

Note: If the output of a Reservoir is specified as being an array (a vector or a matrix), the Upper and Lower Bounds must, by definition, themselves be arrays. Because these fields are optional (you must check the box to activate them), if the box is selected, you cannot leave the field blank (like you can do for the Rate of Change inputs). For example, if you wish to specify a vector of zeroes (e.g., for the lower bound), you must do so by entering “vector(0)”.

How a Reservoir Computes the Overflow Rate

The Overflow_Rate output of a Reservoir has meaning (and is computed) only if you have specified an Upper Bound. GoldSim computes the Overflow_Rate such that the flowing material (e.g., water, widgets, people, rocks) is conserved. To understand how it does so, recall the key assumption regarding the Euler integration method: The specified Rates of Change (Additions and Withdrawal Requests) represent constant rates over the next timestep.

Hence, at the beginning of a timestep, GoldSim can project what the Value will be at the end of timestep, compare it to the Upper Bound, and easily compute an Overflow_Rate for that timestep. To illustrate this, consider the following simple example. Assume we are modeling a pond with an Upper Bound of 3000 m$^3$, and that at time = 10 days, the pond contains 2700 m$^3$ of water. Let’s further assume that our model has a 2 day timestep. Assume that for times greater than 10 days, the Additions/Rate of Change and the Withdrawal Requests/Rate of Change are constant and equal to 500 m$^3$/day and 100 m$^3$/day, respectively, so that the pond will fill up and start to overflow at 10.75 days.

In order to compute overflow rates accurately, by default GoldSim inserts an “unscheduled update” when the Reservoir reaches an Upper Bound. Unscheduled updates are timesteps that are automatically inserted by GoldSim during the simulation in order to more accurately simulate the system.

Read more: Understanding Timestepping in GoldSim (page 455).

In particular, the Overflow_Rate for the example described above is treated as follows. At 10 days, GoldSim would report an (instantaneous) Overflow_Rate of zero. A new timestep (an “unscheduled update”) would then be inserted at 10.75 days when the pond would start to overflow. The Overflow_Rate at this time would actually be internally calculated as 400 m$^3$/day. However, because by default GoldSim only reports values at scheduled timesteps (e.g., 10 days, 12 days), the Overflow_Rate will not be displayed as 400 m$^3$/day until 12 days.

Hence, it is important to understand that the Overflow_Rate represents the instantaneous overflow rate at the reported time. You cannot assume that
**Using Stock Elements**

**Chapter 4: Using the GoldSim Elements**

- **243**

- **Bold**: this rate is constant over the next scheduled timestep. That is, it would be incorrect to assume that the overflow rate was zero between 10 and 12 days. In fact, at 10 days, the overflow rate was indeed 0 m$^3$/day, but it changed to 400 m$^3$/day at 10.75 days (between the scheduled updates). Because 10.75 days is an unscheduled timestep, you would not see this in a time history plot. However, if the Overflow_Rate was integrated (using an Integrator or another Reservoir that represented the total amount of water that has overflowed), the following results would be obtained:

<table>
<thead>
<tr>
<th>Time (days)</th>
<th>Reported Overflow Rate (m$^3$/day)</th>
<th>Total Water Overflowed (m$^3$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>400</td>
<td>500</td>
</tr>
<tr>
<td>14</td>
<td>400</td>
<td>1300</td>
</tr>
</tbody>
</table>

Note that the amount of water is properly conserved. At 12 days, the pond has been overflowing for 1.25 days (at a rate of 400 m$^3$/day), so that a total volume of 500 m$^3$ has overflowed. At 14 days, the pond as been overflowing for 3.25 days, so a total volume of 1300 m$^3$ has overflowed.

**Note**: In some cases, it may be of interest to see the values (such as the Overflow_Rate) that were computed at unscheduled updates. To facilitate this, GoldSim provides an option to do so (under a specified set of conditions) in the Advanced Time settings.

**Read more**: Including Unscheduled Updates in Time History Results (page 477).

**Note**: Most spreadsheet models implicitly compute average values (e.g., flows) over a step (or the change from one step to the next). You can compute average values such as these (e.g., in order to compare GoldSim with a spreadsheet model) by using Goldsim’s Reporting Periods feature.

**Read more**: Defining Reporting Periods (page 461).

**Note**: As noted above, by default, GoldSim inserts an unscheduled update when the Reservoir reaches an Upper Bound. However, if desired, you can disable unscheduled updates (although it is generally not recommended). If you do so, the reported Overflow_Rate is the average rate over the next scheduled timestep and can be assumed to be constant. (This option is accessed via the Advanced… button in the Time tab of the Simulation Setting dialog).

**Read more**: Controlling Unscheduled Updates (page 470).
Note: You can record when Reservoirs hit or depart from their Upper and Lower Bounds in the model’s Run Log. This logging can be activated via the General tab of the Options dialog (accessed via Model|Options from the main menu).

Read more: The General Tab of the Options Dialog (page 446).

The calculation of an Overflow_Rate is further complicated because the Upper Bound can be specified to change with time (i.e., you could specify this input as being time-variable). If you do so, GoldSim makes the following assumption: like the Rates of Change, the Upper Bound is assumed to remain constant over a timestep.

For example, if you specify that the Upper Bound varies linearly from 3000 m$^3$ to 2000 m$^3$ over 100 days, and simulate the system with a 10 day timestep, GoldSim will approximate the Upper Bound curve internally as a stair-step function:

![Diagram showing the approximation of the Upper Bound curve]

Hence, within a given timestep, GoldSim follows the same logic outlined in the previous example to compute the Overflow_Rate for that timestep.

Note that an Overflow_Rate can be generated even if there is no net inflow of material into the Reservoir. This can happen if the Upper Bound decreases below the Value. To illustrate this, consider the following simple example.

Assume we are modeling a pond with an Upper Bound of 3000 m$^3$. Further assume that at the beginning of a timestep, the pond contains 2700 m$^3$ of water. There are no further additions or withdrawals from the pond. However, assume that over the next timestep, the Upper Bound decreases from 3000 m$^3$ to 2500 m$^3$ (e.g., it fills up with silt).

How would GoldSim handle this? During the timestep the Upper Bound would be assumed to remain at 3000 m$^3$. At the beginning of the following timestep, however, it would instantaneously decrease to 2500 m$^3$. This would result in
2700 – 2500 = 200 $m^3$ of water that would instantaneously have to overflow. GoldSim actually spreads this out over the following timestep, such that with a 2 day timestep, the Overflow_Rate over the following timestep would be $200/2 = 100 m^3$/day.

**Note:** If the **Upper Bound** is constant, the Value in the Reservoir will never exceed the **Upper Bound**. However, if the **Upper Bound** is changing with time, under some circumstances, it is possible for the Value of the Reservoir to exceed the **Upper Bound**. If the **Upper Bound** is exceeded (because it has changed during a timestep), GoldSim will overflow the excess uniformly over the next timestep.

**Note:** Although GoldSim conserves mass in these situations, this does not imply that the result is perfectly accurate. In fact, the accuracy of the result is a function of the timestep, since GoldSim is approximating continuously varying functions (i.e., rates and bounds) using stair-step functions. This approximation is discussed further in Appendix F.

---

**How a Reservoir Computes the Withdrawal Rate**

In the absence of a **Lower Bound**, the Withdrawal_Rate output is equal to the Withdrawal Requests/Rate of Change input. However, if a **Lower Bound** is specified, the Withdrawal_Rate output represents the actual rate of withdrawal from the Reservoir, which may be less than the specified Withdrawal Requests/Rate of Change (which can be thought of as the demand or requested rate of withdrawal).

**Note:** The Withdrawal_Rate is completely independent of and does not include the Overflow_Rate. The Withdrawal_Rate only considers of withdrawals specified in the Withdrawal Requests/Rate of Change input field.

GoldSim compares the Withdrawal Requests/Rate of Change to the amount of material available, and only outputs what can actually be withdrawn from the Reservoir over the timestep. Hence, if the Value is above the **Lower Bound** throughout the timestep, the Withdrawal_Rate output is exactly equal to the specified Withdrawal Requests/Rate of Change. If the Value is at the **Lower Bound** throughout the timestep, the Withdrawal_Rate output is equal to the specified Additions/Rate of Change.

If the Reservoir reaches (or leaves) the **Lower Bound** during the timestep, GoldSim uses logic similar to that for computing overflow rates in order to correctly compute the actual withdrawal rate. In order to compute withdrawal rates accurately, by default GoldSim inserts an “unscheduled update” when the Reservoir reaches a **Lower Bound**. Unscheduled updates are timesteps that are automatically inserted by GoldSim during the simulation in order to more accurately simulate the system.

**Read more:** Understanding Timestepping in GoldSim (page 455).

To illustrate how GoldSim treats withdrawal rates, consider the following simple example. Assume we are modeling a pond with a **Lower Bound** of zero. Further assume that at time = 10 day, the pond contains 200 $m^3$ of water. GoldSim computes the constant Additions/Rate of Change and the Withdrawal Requests/Rate of Change for times greater than 10 days to be 100
m³/day and 300 m³/day, respectively. Therefore, the pond will run out of water at 11 days. Let’s further assume that our model has a 2 day timestep (so that this occurs in the middle of the scheduled timesteps).

The Withdrawal_Rate is computed as follows. At 10 days, GoldSim will report a Withdrawal_Rate of 300 m³/day. A new timestep (an “unscheduled update”) is then inserted at 11 days when the pond becomes empty. The Withdrawal_Rate at this time will actually be internally calculated as 100 m³/day (since the pond is empty, the Withdrawal_Rate can be no greater than the Additions/Rate of Change). However, because by default GoldSim only reports values at scheduled timesteps (e.g., 10 days, 12 days), the Withdrawal_Rate will not be displayed as 100 m³/day until 12 days.

Hence, it is important to understand that the Withdrawal_Rate represents the instantaneous withdrawal rate at the reported time. You cannot assume that this rate is constant over the next scheduled timestep. That is, it would be incorrect to assume that the Withdrawal_Rate was 300 m³/day between 10 and 12 days. In fact, at 10 days, the Withdrawal_Rate was indeed 300 m³/day, but it changed to 100 m³/day at 11 days (between the scheduled updates). Because 11 days is an unscheduled timestep, you would not see this in a time history plot. However, if the Withdrawal_Rate was integrated (using an Integrator or another Reservoir that represented the total amount of water that has been withdrawn), the following results would be obtained:

<table>
<thead>
<tr>
<th>Time (days)</th>
<th>Reported Withdrawal Rate (m³/day)</th>
<th>Total Water Withdrawn (m³)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>300</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>100</td>
<td>400</td>
</tr>
<tr>
<td>14</td>
<td>100</td>
<td>600</td>
</tr>
</tbody>
</table>

Note that the amount of water is properly conserved. At 12 days, the pond has been empty for 1 day (and hence the Withdrawal_Rate during that period has been equal to the inflow rate of 100 m³/day), so that a total volume of 300 m³/day * 1 day + 100 m³/day * 1 day = 400 m³ has been withdrawn. At 14 days, the pond as been empty for 3 days, so that a total volume of 300 m³/day * 1 day + 100 m³/day * 3 day = 600 m³ has been withdrawn.

Note: In some cases, it may be of interest to see the values (such as the Withdrawal_Rate) that were computed at unscheduled updates. To facilitate this, GoldSim provides an option to do so (under a specified set of conditions) in the Advanced Time settings.

Read more: Including Unscheduled Updates in Time History Results (page 477).

Note: Most spreadsheet models implicitly compute average values (e.g., flows) over a step (or the change from one step to the next). You can compute average values such as these (e.g., in order to compare GoldSim with a spreadsheet model) by using Goldsim’s Reporting Periods feature.

Read more: Defining Reporting Periods (page 461).
**Note:** As noted above, by default, GoldSim inserts an unscheduled update when the Reservoir reaches a **Lower Bound**. However, if desired, you can disable unscheduled updates (although it is generally not recommended). If you do so, the reported Withdrawal_Rate is the average rate over the next scheduled timestep and can be assumed to be constant. (This option is accessed via the Advanced… button in the Time tab of the Simulation Setting dialog).

*Read more:* [Controlling Unscheduled Updates](page 470).

**Note:** You can record when Reservoirs hit or depart from their **Upper** and **Lower Bounds** in the model’s Run Log. This logging can be activated via the General tab of the Options dialog (accessed via Model|Options from the main menu).

*Read more:* [The General Tab of the Options Dialog](page 446).

**Note:** In many material flow models, you will likely have multiple withdrawals from a Reservoir, with each withdrawal being directed to a different place. For example, if you were simulating a pond, you could imagine having three different withdrawals from the pond: 1) pumping out of the pond (e.g., to another pond); 2) leakage out the bottom pond (into the underlying ground); and 3) evaporation (into the atmosphere). In such a case, the Withdrawal_Rate output would represent the total withdrawal. In such a case, you should use the Pool element, which is an expanded version of the Reservoir specifically designed to handle multiple withdrawals.

*Read more:* [Pool Elements](page 252).

The calculation of a Withdrawal_Rate can be further complicated because the **Lower Bound** can be specified to change with time (i.e., you could specify this input as being time-variable).

**Note:** If the **Lower Bound** is constant, the Value in the Reservoir will never fall below the **Lower Bound**. However, if the **Lower Bound** is changing with time, under some circumstances, it is possible for the Value of the Reservoir to fall below the **Lower Bound**. The primary purpose of the **Lower Bound** is to accurately compute the Withdrawal_Rate. If the **Lower Bound** exceeds the amount in the Reservoir (because the **Lower Bound** has changed during a timestep), the Reservoir will not permit any withdrawals until it returns to the **Lower Bound** (via additions).

As is the case with **Upper Bounds**, GoldSim carries out such calculations by assuming that bounds remain constant over a timestep. A simple example illustrating this is provided in the section discussing how GoldSim computes the Overflow_Rate with a changing **Upper Bound**.

*Read more:* [How a Reservoir Computes the Overflow Rate](page 242).
Note: Although GoldSim conserves mass when computing the actual withdrawal rate, it does not imply that the result is perfectly accurate. In fact, the accuracy of the result is a function of the timestep, since GoldSim is approximating continuously varying functions (i.e., rates and bounds) using stair-step functions. This approximation is discussed further in Appendix F.

Avoiding Oscillatory Behavior When Using Reservoirs

Some types of systems can exhibit oscillatory behavior, showing repetitive variation about a central value or between two or more different states. In some cases, this behavior is real. A pendulum is the classic example of such a system. It is also possible, however, to create logic that results in unrealistic oscillatory behavior. Most commonly, poor logic defining the Addition Rate and the Withdrawal Rate for a Reservoir can cause it to oscillate in an unrealistic manner.

To illustrate this, let’s consider the example pictured below:

In this simple model, water flows into a pond at a constant rate (100 m$^3$/day), and we wish to actively control the volume of water in the pond (maintaining a target volume of 300 m$^3$) by turning a pump (that removes water from the pond) on and off in a specified manner. The pump, when on, pumps at a rate of 200 m$^3$/day.

The Withdrawal from the pond is defined as follows:

\[
\text{Equation} = \begin{cases} 
\text{if} (\text{Pond} \geq \text{Target}, \text{Pumping Rate}, 0 \text{ m}^3/\text{day})
\end{cases}
\]

That is, when the pond volume is greater than or equal to the target volume, the pump is turned on. When it is below the target volume, it is turned off. Note that the Pumping Rate is larger than the Inflow, so when the pump is turned on, the volume in the pond is reduced. This creates a feedback loop between the pond volume and the withdrawal – it is an active feedback control system that is analogous to those common in many engineered systems.

Read more: Evaluating Feedback Loops (page 346).

But will this logic work? At first glance, it seems to make sense, but if you were run the model and look at the results, you would quickly realize that a real system would be unlikely to be controlled by logic like this. The result looks like this:
As you can see, the pond fills up, and then oscillates right around 300 m³. As soon as it reaches the target, the pump turns on, the volume is reduced and the pump turns off, and so on. The timestep for this model is 1 hr, and the model oscillates back and forth every hour. In fact, if we reduced the timestep to 1 minute, as long as the pumping rate is high enough (and it is), it would oscillate every minute! Clearly, the system would never be designed to operate like this in the real world (e.g., the pump would not last very long if it was turned on and off every minute).

In this example, the goal is to control the Reservoir around a particular “target” level. Although this is a common objective (e.g., a thermostat), it is also quite common for feedback control systems like this to be used to control the behavior of a system as it approaches its bounds (and these are the situations most likely to result in oscillatory behavior). In this case, you can imagine the need to actively adjust flows as the volume approaches the bounds. Using similar logic as above, you might choose to turn off the pump when the pond empties and turn it on as soon as it exceeds zero:

Unfortunately, such logic will often cause the model to oscillate even more rapidly than in the previous example. This is because GoldSim inserts unscheduled updates whenever a bound is reached. Logic such as this could result in the Reservoir continuously oscillating at the bound (with a timestep being inserted each time the bound is reached). This can result in timesteps being inserted practically continuously (e.g., every second or less!)

Read more: Understanding Timestepping in GoldSim (page 455).

So what is the proper way to model these kinds of systems? The key is to build logic that is realistic (i.e., logic that could actually be practically implemented in the real world). For example, if you really did want to manage a Reservoir to a target, what you would likely do is use what is referred to as a “deadband” or a target range within which the system is controlled. This is actually how a thermostat typically works (e.g., “turn the heat on when the temperature is 68 degrees and turn it off when it reaches 70 degrees”). You could implement this in GoldSim using a Status element.
Read more: Simulating a Deadband Using a Status Element (page 400).

When trying to control the behavior of a system as it approaches its bounds (as pointed out above, often the situations most likely to result in oscillatory behavior), it becomes particularly important to model the system realistically. In particular, additions/withdrawals should either 1) be changed abruptly before the bound is reached (e.g., a pump turns off when the water level drops to a certain level but before the pond is actually empty); or 2) gradually ramped down such that when the Reservoir reaches its bound, the additions/withdrawals have gradually changed to the appropriate value (e.g., gradually ramp the withdrawal rate down as the volume decreases until it reaches zero when the volume reaches the lower bound). If using the first approach, you would also likely implement a deadband, so that, for example, the pump turns off when the water level drops to a certain level (but before the pond is actually empty), and turns back on at some higher level.

Using the Is_Full Output of a Reservoir

The Is_Full output is only available if you specify an Upper Bound for a Reservoir. It is a condition (False if the Reservoir is below the Upper Bound, and True if it is at the Upper Bound).

This output is useful because it is a special type of output called a state variable (the primary output of a Reservoir is also a state variable). This has the important implication that inputs to the Reservoir (e.g., the Additions/Rate of Change) can reference this output without causing a recursive error.

Read more: Evaluating Feedback Loops (page 346).

As an example, suppose that you wanted to add water to a Reservoir only if it was not overflowing; once it started to overflow, you wanted the flow rate to go to zero. To accomplish this, you could define an Expression for the inflow rate as follows:

\[
\text{if(Reservoir 1.Overflow Rate > 0 m3/day, 0 m3/day, 500 m3/day)}
\]

Unfortunately, because the Overflow_Rate is not a state variable, if you then tried to link this Expression into the Additions/Rate of Change input for the Reservoir, you would get this error:

However, you can solve this problem by using the Is_Full output:

\[
\text{if(Reservoir 1.Is Full, 0 m3/day, 500 m3/day)}
\]

If the Expression was defined as above, then you could link to this Expression into the Additions/Rate of Change field without causing a recursive error.
Note: Although the Is_Full output allows you to create such feedback loops, you need to take care when using such logic to avoid creating unrealistic oscillatory behavior.

Read more: Avoiding Oscillatory Behavior When Using Reservoirs (page 248).

Specifying Discrete Additions and Withdrawals to a Reservoir

In addition to continuous Rates of Addition and Withdrawal Requests, Reservoirs can also accept discrete changes. That is, the Reservoirs actually compute their Value by accounting for discrete changes as follows:

\[ \text{Value}(t_n) = \text{Initial Value} + \sum_{i=1}^{n} \text{Rate of Change}(t_i - \Delta t_i) \Delta t_i + \sum \text{Discrete Additions} - \sum \text{Discrete Withdrawals} \]

Discrete Additions and Withdrawals are specified by checking the Discrete Change box (in the Additions and/or Withdrawal Requests sections) and by specifying a link to a discrete change signal with an Add instruction:

Note that if you check a Discrete Change checkbox, GoldSim will add new outputs to the element (for Additions, Discrete_Overflow if there is an Upper Bound defined, and for Withdrawal Requests, Discrete_Wdrawals).

Instantaneously Replacing the Current Value of a Reservoir

In some cases, you may want to instantaneously replace the current value of a Reservoir with a specified value.

This can be accomplished by checking the Additions field and specifying a link to a discrete change signal with a Replace instruction.

Read more: Basic Concepts of Discrete Event Modeling (page 352); Modeling Discrete Changes to a Reservoir (page 386).

Note: If you have specified an Upper or Lower Bound, and choose to Replace the current value of a Reservoir, the replacement value must be within the bounds (or GoldSim will display a fatal error message).

Browser View of a Reservoir Element

The browser view of a Reservoir element shows up to seven inputs and up to six outputs (depending on which boxes are checked in the dialog). If the Upper and Lower Bound boxes are checked, the browser view looks like this:
Pool Elements

Pool elements are elements that, like Reservoirs, accumulate flows. And like Reservoirs, you can use them to accumulate and track material, such as the quantity of water in a lake, the amount of soil in a stockpile, the number of people in a city, and the number of widgets in a warehouse.

Their default symbol is very similar to that of a Reservoir - a container of water - since this is an excellent analogy for the behavior of the element. Like a container of water, a Pool element has a current value (e.g., the volume of water in the container), inflows, withdrawals and overflows.

Read more: Reservoir Elements (page 235).

A Pool is an expanded (and more powerful) version of the Reservoir element. It can do everything that a Reservoir can do (e.g., it can have Upper and Lower Bounds), but is different (and hence more powerful) in three important ways:

- Pools allow you to specify Inflows as separate and individual inputs. (Reservoirs provide only a single input for inflows).
- Pools allow you to specify Requested Outflows as separate and individual inputs. (Reservoirs provide only a single input for a “withdrawal request”). In turn, in addition to providing a “total outflow” output, Pools provide separate and individual outputs for each requested outflow. (Reservoirs provide only a single “withdrawal rate” output). This allows you to easily route different outflows to different locations.
- Because Pools allow you to specify multiple Requested Outflows, and it is possible (if the Pool reaches its Lower Bound) that the Pool may not be able to provide the total for all requests, you also specify Priorities for each requested outflow, and the total outflow is then allocated accordingly among the requests if the Pool is unable to provide all of the requests.

Note: Due to the differences pointed out above, if you are modeling a system that has multiple distinct outflows, you should use a Pool rather than a Reservoir.
The fundamental inputs to a Pool are an **Initial Quantity**, **Inflows** and **Outflows**.

In the absence of Upper and Lower Bounds, the primary output of the Pool (its **Quantity**) is computed as follows:

\[
\text{Quantity} = \text{Initial Quantity} + \int (\text{Inflows} - \text{Outflows}) \, dt
\]

The Inflows and Outflows can, of course, be functions of time.

The Properties dialog for a Pool element looks like this:

The Pool dialog has three tabs: a **Definition** tab (where the **Units**, **Initial Quantity**, **Lower Bound**, **Upper Bound** and discrete **Additions** and **Withdrawals** are specified), an **Inflows** tab (where inflows to the Pool are specified), and an **Outflows** tab (where requested outflows from the Pool are specified).

All Pools have the following outputs:

- The quantity in the Pool (the primary output, having the same name as the element);
- The **Total_Inflow** to the Pool (simply the sum of all the Inflow inputs);
- The **Total_Request** from the Pool (simply the sum of all the Requested Outflow inputs);
- The **Total_Outflow** from the Pool (which is always less than or equal to the **Total_Request**); and
- Each individual Outflow from the Pool (whose names are user-defined).

Optional outputs include:

- The **Overflow** from the Pool (only available if an Upper Bound is defined);
- **Is_Full**, a condition indicating whether or not the Pool is at its Upper Bound (only available if an Upper Bound is defined);
Using Stock Elements

- Discrete_Overflow, a discrete change signal (only available if an Upper Bound is defined and a Discrete Addition is specified); and
- Discrete_Withdrawals, a discrete change signal (only available if a Discrete Withdrawal is specified).

An example model which illustrates the use of Pool elements (Pool.gsm) can be found in the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

Within the Properties dialog, two sets of Units must be specified for a Pool (the Quantity Units and the Flow Units):

The primary output of a Pool is its Current Value (its Quantity). The Quantity Units determine the dimensions of this output. This output is always a value (it cannot be a condition). Moreover, it can only be a scalar (and hence, all of the inputs for the element must be scalars).

**Note:** If you need to model a vector or matrix quantity, you must use a Reservoir element.

The Flow Units determine the display units for the Outflow outputs of the element. The dimensions of this output must be the same as those of the Quantity divided by Time. For example, if the Quantity Units have dimensions of Volume, the Flow Units must have dimensions of Volume/Time.

**Note:** GoldSim encourages (but does not require) you to specify Quantity Units for a Pool whose dimensions are consistent with materials. In particular, it expects the units to have dimensions of mass, volume, energy, amount, currency, or to be dimensionless. When you first specify your Quantity Units, GoldSim will warn you if your units do not have one of these dimensions. GoldSim will not, however, prevent you from using other units.

The Initial Quantity input to the Pool must have the same dimensions at the Quantity Units.

**Note:** The Initial Quantity must be a number or a link from a static variable (e.g., a constant Data element or a Stochastic).

Specifying the Inflow Rates for a Pool

The inflows to a Pool are specified using the Inflows tab:
You can add **Inflows** by pressing the **Add** button. A browser tree showing all of the elements in the model will be displayed. Note that this tree is organized by *containment* in the same manner as the main browser. To insert a link from this dialog, you select a specific output (or an element having a primary output), and then press **OK**. If you press **Cancel**, GoldSim will insert a new blank (zero) item. In either case, the dialog will close and the item will be added to the list of Inflows:

**Inflows** can be deleted using the **Remove** button moved up and down in the list using the two buttons to the right of the **Remove** button.

**Inflows** can be specified as constants or expressions and/or links, and may change with time. All **Inflows** must be non-negative values. Negative values for these inputs during a simulation will result in a fatal error.
Note: The specified Inflows represent constant rates over the next timestep. Hence, if an Inflow was defined as “if(time > 10 day, 2 m³/day, 1 m³/day)”, and you were using a 1 day timestep, the rate would not actually change to 2 m³/day until time = 11 days. That is, since at time = 10 days, the if statement indicates that the rate is equal to 1 m³/day, GoldSim would assume that the rate was equal to 1 m³/day between 10 days and 11 days. If you wanted the rate to change at 10 days, you would write the if statement as “if(time >= 10 day, 2 m³/day, 1 m³/day)”.

Specifying the Outflow Requests for a Pool

The outflow requests from a Pool are specified using the Outflows tab:

By default, a Pool element initially has one Outflow Request (named Outflow1). You can add additional Outflow Requests by pressing the Add button. Outflows can be deleted using the Remove button, and moved up and down in the list using the two buttons to the right of the Remove button.

Note: You cannot delete an Outflow Request if it is the only one (there must always be one). Of course, it can be set to zero (which is the default).

The names of the Outflows can be changed by editing the items in the Name column.

For each Outflow Request in the list, you specify the actual Request. The Request has the same dimensions as the Flow Units.

Read more: Specifying the Dimensions and Initial Quantity for a Pool (page 254).

Requests can be specified as constants or expressions and/or links, and may change with time.
Note: The specified Requests represent constant rates over the next timestep. Hence, if a Request was defined as “if(time > 10 day, 2 m$^3$/day, 1 m$^3$/day)”, and you were using a 1 day timestep, the Request would not actually change to 2 m$^3$/day until time = 11 days. That is, since at time = 10 days, the if statement indicates that the Request is equal to 1 m$^3$/day, GoldSim would assume that the Request was equal to 1 m$^3$/day between 10 days and 11 days. If you wanted the Request to change at 10 days, you would write the if statement as “if(time >= 10 day, 2 m$^3$/day, 1 m$^3$/day)”.

GoldSim sums the Outflow Requests to compute the Total_Request output. If the Pool stays above the Lower Bound (or there is no specified Lower Bound), then the Total_Outflow output is equal to the Total_Request. If the Pool reaches the Lower Bound, however, GoldSim compares the Total_Request to the amount of material available, and the Total_Outflow is computed such that it represents what can actually be withdrawn from the Pool at that time (which, depending on the Inflows, may be less than the Total_Request).

Read more: How a Pool Computes the Total Outflow (page 263).

Note that if the Total_Outflow is limited (such that it is less than the Total_Request), and there are multiple Outflow Requests, it becomes necessary to allocate the Total_Outflow among the requests. This is done based on the specified Priority for each Outflow Request.

Read more: How a Pool Computes the Individual Outflows (page 265).

Numerically, GoldSim approximates the integral represented by a Pool as a sum:

\[
\text{Quantity}(t_n) = \text{Initial Quantity} + \sum_{i=1}^{n} \left[ \text{Inflows}(t_i - \Delta t_i) - \text{Outflows}(t_i - \Delta t_i) \right] \Delta t_i
\]

In these equations, \( \Delta t_i \) is the timestep length just prior to time \( t_i \) (typically this will be constant in the simulation), and \( \text{Quantity}(t_n) \) is the value at end of timestep \( n \). Note that the Quantity at time \( t_i \) is a function of the Inflows and Outflows at previous timesteps (but is not a function of these flows at time \( t_i \)).

Note that if you specify Upper or Lower Bounds for a Pool, the equation shown above is constrained by the specified bounds (i.e., the Quantity cannot exceed the Upper Bound and can not be less than the Lower Bound).

Read more: Defining Upper and Lower Bounds for a Pool (page 258).

Pools, Reservoirs and Integrators use the same numerical integration method, referred to as Euler Integration. A simple example illustrating this integration method is provided in the section discussing how an Integrator element computes its output.

Read more: How an Integrator Element Computes its Output (page 229).

The key assumption involved in this integration method is that at any point in time, the rates (Inflows and Outflow Requests) represent the rates over the next timestep, and those rates are assumed to be constant over the timestep. Euler Integration is discussed in additional detail in Appendix F.
**Note:** Euler integration is the simplest and most common method for numerical integration. However, if the timestep is large, this method can lead to significant errors. This is particularly important for certain kinds of systems in which the error is cumulative (e.g., sustained oscillators such as a pendulum). In these cases, these errors can be reduced by using Containers with Internal Clocks, which allow you to locally use a much smaller timestep.

---

**Defining Upper and Lower Bounds for a Pool**

By default, when you create a new Pool element, a **Lower Bound** of 0 is assumed. You can subsequently remove the **Lower Bound**, or modify it.

Checking one or both of these boxes, however, provides additional functionality and has important impacts on the outputs for the element. In particular:

- If the **Lower Bound** is checked, GoldSim enforces that limitation on the Quantity. As a result, the **Total_Outflow** output is not necessarily equal to the **Total_Request** output (which simply sums the **Outflow Requests**). In this case, the **Total_Outflow** output represents the actual outflow from the Pool, which may be less than the specified sum of the **Outflow Requests** (which represents the demand or requested outflow).

  For example, if the Pool is at the **Lower Bound**, and there are no **Inflows**, the **Total_Outflow** output must be equal to zero, regardless of the **Total_Request** (there is no material available to outflow). Similarly, if the Pool is at the **Lower Bound**, and the **Inflows** total to 10, the **Total_Outflow** output can be no greater than 10, regardless of the **Total_Request** (if there is no storage, you cannot outflow more than is being supplied via inflow).
Note: If the **Lower Bound** is constant, the Quantity in the Pool will never fall below the **Lower Bound**. However, if the **Lower Bound** is changing with time, under some circumstances, it is possible for the Quantity in the Pool to fall below the **Lower Bound**. The primary purpose of the **Lower Bound** is to accurately compute the Total_Outflow. If the **Lower Bound** exceeds the amount in the Pool (because the **Lower Bound** has changed during a timestep), the Pool will not permit any outflows until it returns to the **Lower Bound** (via inflows).

- If the **Upper Bound** is checked, GoldSim enforces that limitation on the Quantity. GoldSim also adds two new (secondary) outputs: Overflow (a value) and Is_Full (a condition). If the Quantity is below the **Upper Bound**, the Overflow output is equal to zero and the Is_Full output is False. If the Quantity is at the **Upper Bound**, the Overflow output is equal to the difference between the sum of the specified Inflows and the sume of the specified Requested Outflows and the Is_Full output is True.

**Read more:** [Using the Is_Full Output of a Pool](page 272).

Note: If the **Upper Bound** is constant, the Quantity in the Pool will never exceed the **Upper Bound**. However, if the **Upper Bound** is changing with time, under some circumstances, it is possible for the Quantity in the Pool to exceed the **Upper Bound**. If the **Upper Bound** is exceeded (because it has changed during a timestep), GoldSim will overflow the excess uniformly over the next timestep.

Note: If your specified **Upper Bound** is reached during a simulation, and the Overflow output is not referenced by any other element (e.g., as an inflow to another Pool), GoldSim will display a warning, as this could indicate that you have neglected to track flows leaving the Pool.

The Pool element is particularly powerful because it enforces the bounds while properly conserving the material being tracked. That is, if a Pool’s Inflows cause it to reach its **Upper Bound**, GoldSim produces an Overflow (which can be routed, for example, to a downstream Pool). Similarly, if the sum of the Outflow Requests exceeds what can actually be delivered, GoldSim accurately reports the actual Total_Outflow.

Furthermore, as is discussed below, the **Upper** and **Lower Bounds** can be specified as functions of time. This allows you to simulate systems such as a pond which is filling up with silt and sand (and hence decreasing its capacity), and a warehouse whose capacity is being expanded (via construction) throughout a simulation.

The **Lower Bound** and the **Upper Bound** inputs to the Pool must have the same dimensions as the Pool’s **Quantity Units**. In addition, the following limitations imposed on the inputs should be specifically noted:

- The **Upper Bound** must be greater than or equal to the **Lower Bound**.
• The Initial Quantity must be greater than or equal to the Lower Bound and less than or equal to the Upper Bound.

The Overflow output of a Pool has meaning (and is computed) only if you have specified an Upper Bound. GoldSim computes the Overflow such that the flowing material (e.g., water, widgets, people, rocks) is conserved. To understand how it does so, recall the key assumption regarding the Euler integration method: The specified Rates (Inflows and Outflow Requests) represent constant rates over the next timestep.

Hence, at the beginning of a timestep, GoldSim can project what the Quantity will be at the end of timestep, compare it to the Upper Bound, and easily compute an Overflow for that timestep. To illustrate this, consider the following simple example. Assume we are modeling a pond with an Upper Bound of 3000 m$^3$, and that at time = 10 days, the pond contains 2700 m$^3$ of water. Let’s further assume that our model has a 2 day timestep. Assume that for times greater than 10 days, the Inflows and the Outflow Requests are constant and equal to 500 m$^3$/day and 100 m$^3$/day, respectively, so that the pond will fill up and start to overflow at 10.75 days.

In order to compute overflow rates accurately, by default GoldSim inserts an “unscheduled update” when the Pool reaches an Upper Bound. Unscheduled updates are timesteps that are automatically inserted by GoldSim during the simulation in order to more accurately simulate the system.

Read more: Understanding Timestepping in GoldSim (page 455).

In particular, the Overflow for the example described above is treated as follows. At 10 days, GoldSim would report an (instantaneous) Overflow of zero. A new timestep (an “unscheduled update”) would then be inserted at 10.75 days when the pond would start to overflow. The Overflow at this time would actually be internally calculated as 400 m$^3$/day. However, because by default GoldSim only reports values at scheduled timesteps (e.g., 10 days, 12 days), the Overflow will not be displayed as 400 m$^3$/day until 12 days.

Hence, it is important to understand that the Overflow represents the instantaneous overflow rate at the reported time. You cannot assume that this rate is constant over the next scheduled timestep. That is, it would be incorrect to assume that the Overflow was zero between 10 and 12 days. In fact, at 10 days, the Overflow was indeed 0 m$^3$/day, but it changed to 400 m$^3$/day at 10.75 days (between the scheduled updates). Because 10.75 days is an unscheduled timestep, you would not see this in a time history plot. However, if the Overflow was integrated (e.g., using an Integrator or a Reservoir that represented the total amount of water that has overflowed), the following results would be obtained:

<table>
<thead>
<tr>
<th>Time (days)</th>
<th>Reported Overflow Rate (m$^3$/day)</th>
<th>Total Water Overflowed (m$^3$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>400</td>
<td>500</td>
</tr>
<tr>
<td>14</td>
<td>400</td>
<td>1300</td>
</tr>
</tbody>
</table>

Note that the amount of water is properly conserved. At 12 days, the pond has been overflowing for 1.25 days (at a rate of 400 m$^3$/day), so that a total volume
of 500 m³ has overflowed. At 14 days, the pond has been overflowing for 3.25 days, so a total volume of 1300 m³ has overflowed.

**Note:** In some cases, it may be of interest to see the output values (such as the Overflow) that were computed at unscheduled updates. To facilitate this, GoldSim provides an option to do so (under a specified set of conditions) in the Advanced Time settings.

**Read more:** Including Unscheduled Updates in Time History Results (page 477).

**Note:** Most spreadsheet models implicitly compute *average* values (e.g., flows) over a step (or the change from one step to the next). You can compute average values such as these (e.g., in order to compare GoldSim with a spreadsheet model) by using Goldsim’s Reporting Periods feature.

**Read more:** Defining Reporting Periods (page 461).

**Note:** As noted above, by default, GoldSim inserts an unscheduled update when the Pool reaches an Upper Bound. However, if desired, you can disable unscheduled updates (although it is generally not recommended). If you do so, the reported Overflow is the average rate over the next scheduled timestep and can be assumed to be constant. (This option is accessed via the Advanced… button in the Time tab of the Simulation Setting dialog).

**Read more:** Controlling Unscheduled Updates (page 470).

**Note:** You can record when Pools hit or depart from their Upper and Lower Bounds in the model’s Run Log. This logging can be activated via the General tab of the Options dialog (accessed via Model|Options from the main menu).

**Read more:** The General Tab of the Options Dialog (page 446).

The calculation of the Overflow is further complicated because the Upper Bound can be specified to change with time (i.e., you could specify this input as being time-variable). If you do so, GoldSim makes the following assumption: like the Inflows and Outflow Requests, the Upper Bound is assumed to remain constant over a timestep.

For example, if you specify that the Upper Bound varies linearly from 3000 m³ to 2000 m³ over 100 days, and simulate the system with a 10 day timestep, GoldSim will approximate the Upper Bound curve internally as a stair-step function:
Hence, within a given timestep, GoldSim follows the same logic outlined in the previous example to compute the Overflow for that timestep.

Note that an Overflow can be generated even if there is no net inflow of material into the Pool. This can happen if the Upper Bound decreases below the value of the Quantity. To illustrate this, consider the following simple example. Assume we are modeling a pond with an Upper Bound of 3000 m$^3$. Further assume that at the beginning of a timestep, the pond contains 2700 m$^3$ of water. There are no further inflows or outflows from the pond. However, assume that over the next timestep, the Upper Bound decreases from 3000 m$^3$ to 2500 m$^3$ (e.g., it fills up with silt).

How would GoldSim handle this? During the timestep the Upper Bound would be assumed to remain at 3000 m$^3$. At the beginning of the following timestep, however, it would instantaneously decrease to 2500 m$^3$. This would result in $2700 - 2500 = 200$ m$^3$ of water that would instantaneously have to overflow. GoldSim actually spreads this out over the following timestep, such that with a 2 day timestep, the Overflow over the following timestep would be $200/2 = 100$ m$^3$/day.

**Note:** If the Upper Bound is constant, the Quantity in the Pool will never exceed the Upper Bound. However, if the Upper Bound is changing with time, under some circumstances, it is possible for the Quantity of the Pool to exceed the Upper Bound. If the Upper Bound is exceeded (because it has changed during a timestep), GoldSim will overflow the excess uniformly over the next timestep.
**How a Pool Computes the Total Outflow**

In the absence of a **Lower Bound**, the **Total_Outflow** output is equal to the **Total_Request** output (which simply sums the **Outflow Requests**). However, if a **Lower Bound** is specified, the **Total_Outflow** output represents the actual rate of outflow from the Pool, which may be less than the specified sum of the **Outflow Requests** (which can be thought of as the demand or requested rate of withdrawal).

**Note:** The **Total_Outflow** is completely independent of and does not include the Overflow. The **Total_Outflow** only only considers outflows resulting from the **Outflow Requests** on the **Outflow** tab.

GoldSim compares the sum of the **Outflow Requests** to the amount of material available, and only outflows what can actually be withdrawn from the Pool over the timestep. Hence, if the Quantity is above the **Lower Bound** throughout the timestep, the **Total_Outflow** output is exactly equal to the sum of the specified **Outflow Requests**. If the Quantity is at the **Lower Bound** throughout the timestep, the **Total_Outflow** output is equal to the sum of the specified **Inflows**.

If the Pool reaches (or leaves) the **Lower Bound** during the timestep, GoldSim uses logic similar to that for computing overflow rates in order to correctly compute the actual **Total_Outflow**. In order to compute the **Total_Outflow** accurately, by default GoldSim inserts an “unscheduled update” when the Pool reaches a **Lower Bound**. Unscheduled updates are timesteps that are automatically inserted by GoldSim during the simulation in order to more accurately simulate the system.

**Read more:** [Understanding Timestepping in GoldSim](#) (page 455).

To illustrate how GoldSim treats outflow rates, consider the following simple example. Assume we are modeling a pond with a **Lower Bound** of zero. Further assume that at time = 10 day, the pond contains 200 m$^3$ of water. GoldSim computes the constant **Inflows** and the **Outflow Requests** for times greater than 10 days to be 100 m$^3$/day and 300 m$^3$/day, respectively. Therefore, the pond will run out of water at 11 days. Let’s further assume that our model has a 2 day timestep (so that this occurs in the middle of the scheduled timesteps).

The **Total_Outflow** is computed as follows. At 10 days, GoldSim will report a **Total_Outflow** of 300 m$^3$/day. A new timestep (an “unscheduled update”) is then inserted at 11 days when the pond becomes empty. The **Total_Outflow** at this time will actually be internally calculated as 100 m$^3$/day (since the pond is empty, the **Total_Outflow** can be no greater than the sum of the **Inflows**. However, because by default GoldSim only reports values at scheduled timesteps (e.g., 10 days, 12 days), the **Total_Outflow** will not be displayed as 100 m$^3$/day until 12 days.

**Hence, it is important to understand that the **Total_Outflow** represents the instantaneous outflow rate at the reported time. You cannot assume that this rate is constant over the next scheduled timestep.** That is, it would be incorrect to assume that the **Total_Outflow** was 300 m$^3$/day between 10 and 12 days.
days. In fact, at 10 days, the Total_Outflow was indeed 300 m$^3$/day, but it changed to 100 m$^3$/day at 11 days (between the scheduled updates). Because 11 days is an unscheduled timestep, you would not see this in a time history plot. However, if the Total_Outflow was integrated (using an Integrator or a Reservoir that represented the total amount of water that had outflowed), the following results would be obtained:

<table>
<thead>
<tr>
<th>Time (days)</th>
<th>Reported Total Outflow (m$^3$/day)</th>
<th>Total Water Outflowed (m$^3$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>300</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>100</td>
<td>400</td>
</tr>
<tr>
<td>14</td>
<td>100</td>
<td>600</td>
</tr>
</tbody>
</table>

Note that the amount of water is properly conserved. At 12 days, the pond has been empty for 1 day (and hence the Total_Outflow during that period has been equal to the inflow rate of 100 m$^3$/day), so that a total volume of 300 m$^3$/day * 1 day + 100 m$^3$/day * 1 day = 400 m$^3$ has outflowed. At 14 days, the pond as been empty for 3 days, so that a total volume of 300 m$^3$/day * 1 day + 100 m$^3$/day * 3 day = 600 m$^3$ has outflowed.

**Note:** In some cases, it may be of interest to see the values (such as the Total_Outflow) that were computed at unscheduled updates. To facilitate this, GoldSim provides an option to do so (under a specified set of conditions) in the Advanced Time settings.

**Read more:** Including Unscheduled Updates in Time History Results (page 477).

**Note:** Most spreadsheet models implicitly compute average values (e.g., flows) over a step (or the change from one step to the next). You can compute average values such as these (e.g., in order to compare GoldSim with a spreadsheet model) by using Goldsim’s Reporting Periods feature.

**Read more:** Defining Reporting Periods (page 461).

**Note:** As noted above, by default, GoldSim inserts an unscheduled update when the Pool reaches a Lower Bound. However, if desired, you can disable unscheduled updates (although it is generally not recommended). If you do so, the reported Total_Outflow is the average rate over the next scheduled timestep and can be assumed to be constant. (This option is accessed via the Advanced… button in the Time tab of the Simulation Setting dialog).

**Read more:** Controlling Unscheduled Updates (page 470).

**Note:** You can record when Pools hit or depart from their Upper and Lower Bounds in the model’s Run Log. This logging can be activated via the General tab of the Options dialog (accessed via Model|Options from the main menu).
Read more: The General Tab of the Options Dialog (page 446).

The calculation of a Total_Outflow can be further complicated because the Lower Bound can be specified to change with time (i.e., you could specify this input as being time-variable).

Note: If the Lower Bound is constant, the Quantity in the Pool will never fall below the Lower Bound. However, if the Lower Bound is changing with time, under some circumstances, it is possible for the Quantity of the Pool to fall below the Lower Bound. The primary purpose of the Lower Bound is to accurately compute the Total_Outflow. If the Lower Bound exceeds the amount in the Pool (because the Lower Bound has changed during a timestep), the Pool will not permit any outflows until it returns to the Lower Bound (via additions).

As is the case with Upper Bounds, GoldSim carries out such calculations by assuming that bounds remain constant over a timestep. A simple example illustrating this is provided in the section discussing how GoldSim computes the Overflow with a changing Upper Bound.

Read more: How a Pool Computes the Overflow (page 260).

Note: Although GoldSim conserves mass when computing the actual outflow, it does not imply that the result is perfectly accurate. In fact, the accuracy of the result is a function of the timestep, since GoldSim is approximating continuously varying functions (i.e., rates and bounds) using stair-step functions. This approximation is discussed further in Appendix F.

How a Pool Computes the Individual Outflows

The Outflow Requests from a Pool are specified using the Outflows tab:

GoldSim sums the Outflow Requests to compute the Total_Request output. If the Pool stays above the Lower Bound (or there is no specified Lower Bound), then the Total_Outflow output is equal to the Total_Request. If the Pool reaches the Lower Bound, however, GoldSim compares the Total_Request to the
amount of material available, and the Total_Outflow is computed such that it represents what can actually be withdrawn from the Pool at that time (which, depending on the Inflows, may be less than the Total_Request).

Read more: How a Pool Computes the Total Outflow (page 263).

Note that if the Total_Outflow is limited (such that it is less than the Total_Request), and there are multiple Outflow Requests, it becomes necessary to allocate the Total_Outflow among the requests. This is done based on the specified Priority for each Outflow Request. Outflow Requests with lower valued Priorities are met first (i.e., Priority 1 is met before Priority 2).

To illustrate this, consider the following example:

The Total_Request output (the sum of the individual Outflow Request inputs) is 130 m$^3$/day.

Let’s further assume that throughout the simulation, the Pool is always above its Lower Bound. In this simple case, the Total_Outflow output would be the same as the Total_Request output (130 m$^3$/day). As a result, the Priority for each Request would never need to be considered (each individual outflow would receive its entire Request).

The outputs for this Pool would look like this:
Note that the “Outflows” folder contains each of the individual outflows.

Now let’s complicate this example a bit more. Let’s assume that the Pool is empty (at its Lower Bound), but the Inflows to the Pool sum to 100 m$^3$/day. In this case, since the Total_Request is equal to 130 m$^3$/day, the Pool would remain empty (more is requested than in coming in, and there is nothing available from storage). Moreover, the Total_Outflow would be limited to 100 m$^3$/day. As a result, it is necessary to allocate the Total_Outflow among the requests. This is done based on the specified Priority for each Outflow Request. Outflow Requests with lower valued Priorities are met first (i.e., Priority 1 is met before Priority 2).

In this case, Outflow1 would have a value of 75 m$^3$/day (its entire request would be met), Outflow2 would have a value of 20 m$^3$/day (its entire request would be met), Outflow3 would have a value of 5 m$^3$/day (only part of its request would be met), and Outflow4 would have a value of 0 m$^3$/day (none of its request would be met).

By default, each output has a fixed (and unique) Priority (which is an integer). The Priority of the outputs always decrease downward (with the first item having a Priority of 1). Priority 1 therefore represents the highest priority demand, 2 the next highest, and so on. You can move outflows up and down the Priority list using the Move Up and Move Down buttons.

If you check the Define Priorities checkbox, the Priority column becomes editable. In this case, the Priorities can be specified as constants or expressions and/or links, and may change with time. They do not need to be integers and can be any real value (positive or negative). Outflows with lower valued Priorities are met first (e.g., Priority -2.4 is met before Priority 5.7).

You can also assigned the same Priority to multiple Outflow Requests. If multiple Outflow Requests have the same Priority, you can specify the manner in which equal priorities are treated in the If equal field. There are two options: 1) “share equally”; or 2) “share proportional to requests”.

If “share equally” (the default) is selected, each Outflow Request with equal Priority receives the same amount up until either its request is satisfied, or nothing more is available. The calculation is carried out recursively starting with the smallest request.

To illustrate this, consider the following example:

...
Again, let’s assume that the Pool is empty (at its Lower Bound), but the Inflows to the Pool sum to 100 m$^3$/day. In this case, since the Total_Request is equal to 110 m$^3$/day, the Pool would remain empty (more is requested than in coming in, and there is nothing available from storage). Moreover, the Total_Outflow would be limited to 100 m$^3$/day. As a result, it is necessary to allocate the Total_Outflow among the requests. Note, however, that the final three requests all have the same Priority.

In this case, Outflow1 would have a value of 75 m$^3$/day (its entire request would be met), Outflow2, Outflow3 and Outflow4 would share the remainder (25 m$^3$/day) as follows:

- Outflow4 would receive its full request (5 m$^3$/day); Outflow2 and Outflow3 would receive the same amount (5 m$^3$/day). This leaves 10 m$^3$/day to be allocated.
- Outflow2 and Outflow3 would then equally share the remaining 10 m$^3$/day (such that each would receive a total of 10 m$^3$/day). So Outflow3 and Outflow4 would meet their demands, and Outflow2 would not.

If “share proportional to requests” is selected, if there is not enough supply to meet all of the requests, it is shared proportionally according to the request. To illustrate this, consider the same example as above, with proportional sharing:
In this case, Outflow1 would still have a value of 75 m$^3$/day (its entire request would be met), Outflow2, Outflow3 and Outflow4 would share the remainder (25 m$^3$/day) as follows:

- Outflow2 would receive 20/35 = 57.1% of the remaining 25 m$^3$/day.
- Outflow3 would receive 10/35 = 28.6% of the remaining 25 m$^3$/day.
- Outflow4 would receive 5/35 = 14.3% of the remaining 25 m$^3$/day.

The table below summarizes the results for these two examples (note that the Total_Inflow (and Total_Outflow) is 100 m$^3$/day and the Total_Request is 110 m$^3$/day):

<table>
<thead>
<tr>
<th>Outflow</th>
<th>Priority</th>
<th>Request (m$^3$/day)</th>
<th>Result with equal sharing (m$^3$/day)</th>
<th>Result with proportional sharing (m$^3$/day)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outflow1</td>
<td>1</td>
<td>75</td>
<td>75</td>
<td>75</td>
</tr>
<tr>
<td>Outflow2</td>
<td>2</td>
<td>20</td>
<td>10</td>
<td>14.3</td>
</tr>
<tr>
<td>Outflow3</td>
<td>2</td>
<td>10</td>
<td>10</td>
<td>7.1</td>
</tr>
<tr>
<td>Outflow4</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>3.6</td>
</tr>
</tbody>
</table>

**Note:** The algorithm described above is the same algorithm used by the Allocator element. In fact, a Pool can be thought of as a combination of a Reservoir and an Allocator.

### Avoiding Oscillatory Behavior When Using Pools

Some types of systems can exhibit oscillatory behavior, showing repetitive variation about a central value or between two or more different states. In some cases, this behavior is real. A pendulum is the classic example of such a system. It is also possible, however, to create logic that results in *unrealistic* oscillatory behavior.
behavior. Most commonly, poor logic defining the Inflows and the Outflow Requests for a Pool can cause it to oscillate in an unrealistic manner.

To illustrate this, let’s consider the example pictured below:

![Diagram of water flow into a pond with inflow and outflow requests.](image)

In this simple model, water flows into a pond at a constant rate (100 m³/day), and we wish to actively control the volume of water in the pond (maintaining a target volume of 300 m³) by turning a pump (that removes water from the pond) on and off in a specified manner. The pump, when on, pumps at a rate of 200 m³/day.

The (single) Outflow Request from the pond is defined as follows:

```
Equation
if(Pond >= Target, Pumping_Rate, 0 m³/day)
```

That is, when the pond volume is greater than or equal to the target volume, the pump is turned on. When it is below the target volume, it is turned off. Note that the Pumping_Rate is larger than the Inflow, so when the pump is turned on, the volume in the pond is reduced. This creates a feedback loop between the pond volume and the outflow – it is an active feedback control system that is analogous to those common in many engineered systems.

**Read more:** Evaluating Feedback Loops (page 346).

But will this logic work? At first glance, it seems to make sense, but if you were run the model and look at the results, you would quickly realize that a real system would be unlikely to be controlled by logic like this. The result looks like this:
As you can see, the pond fills up, and then oscillates right around 300 m³. As soon as it reaches the target, the pump turns on, the volume is reduced and the pump turns off, and so on. The timestep for this model is 1 hr, and the model oscillates back and forth every hour. In fact, if we reduced the timestep to 1 minute, as long as the pumping rate is high enough (and it is), it would oscillate every minute! Clearly, the system would never be designed to operate like this in the real world (e.g., the pump would not last very long if it was turned on and off every minute).

In this example, the goal is to control the Pool around a particular “target” level. Although this is a common objective (e.g., a thermostat), it is also quite common for feedback control systems like this to be used to control the behavior of a system as it approaches its bounds (and these are the situations most likely to result in oscillatory behavior). In this case, you can imagine the need to actively adjust flows as the volume approaches the bounds. Using similar logic as above, you might choose to turn off the pump when the pond empties and turn it on as soon as it exceeds zero:

\[
\text{Equation} \quad \begin{cases} \text{If} \,(\text{Pond} > 0 \text{ m}^3, \text{Pumping Rate}, 0 \text{ m}^3/\text{day}) \end{cases}
\]

Unfortunately, such logic will often cause the model to oscillate even more rapidly than in the previous example. This is because GoldSim inserts unscheduled updates whenever a bound is reached. Logic such as this could result in the Pool continuously oscillating at the bound (with a timestep being inserted each time the bound is reached). This can result in timesteps being inserted practically continuously (e.g., every second or less)!

Read more: Understanding Timestepping in GoldSim (page 455).

So what is the proper way to model these kinds of systems? The key is to build logic that is realistic (i.e., logic that could actually be practically implemented in the real world). For example, if you really did want to manage a Pool to a target, what you would likely do is use what is referred to as a “deadband” or a target range within which the system is controlled. This is actually how a thermostat typically works (e.g., “turn the heat on when the temperature is 68 degrees and turn it off when it reaches 70 degrees”). You could implement this in GoldSim using a Status element.
When trying to control the behavior of a system as it approaches its bounds (as pointed out above, often the situations most likely to result in oscillatory behavior), it becomes particularly important to model the system realistically. In particular, inflows and outflows should either 1) be changed abruptly before the bound is reached (e.g., a pump turns off when the water level drops to a certain level but before the pond is actually empty); or 2) gradually ramped down such that when the Pool reaches its bound, the inflows and outflows have gradually changed to the appropriate value (e.g., gradually ramp the outflow rate down as the volume decreases until it reaches zero when the volume reaches the lower bound). If using the first approach, you would also likely implement a deadband, so that, for example, the pump turns off when the water level drops to a certain level (but before the pond is actually empty), and turns back on at some higher level.

The Is_Full output is only available if you specify an Upper Bound for a Pool. It is a condition (False if the Pool is below the Upper Bound, and True if it is at the Upper Bound).

This output is useful because it is a special type of output called a state variable (the primary output of a Pool is also a state variable). This has the important implication that inputs to the Pool (e.g., the Inflows) can reference this output without causing a recursive error.

As an example, suppose that you wanted to add water to a Pool only if it was not overflowing; once it started to overflow, you wanted the inflow rate to go to zero. To accomplish this, you could define an Expression for the Inflow as follows:

Unfortunately, because the Overflow output is not a state variable, if you then tried to link this Expression into an Inflows field for the Pool, you would get this error:

However, you can solve this problem by using the Is_Full output:

If the Expression was defined as above, then you could link to this Expression into an Inflows field for the Pool without causing a recursive error.
Note: Although the Is_Full output allows you to create such feedback loops, you need to take care when using such logic to avoid creating unrealistic oscillatory behavior.

Specifying Discrete Additions and Withdrawals to a Pool

Read more: Avoiding Oscillatory Behavior When Using Reservoirs (page 248).

In addition to continuous Inflows and Outflow Requests, Pools can also accept discrete changes. In particular, Pools actually compute their Quantity by accounting for discrete changes as follows:

\[
\text{Quantity}(t_n) = \text{Initial Quantity} + \sum_{i=1}^{n} \left[ \text{Inflows}(t_i - \Delta t_i) - \text{Outflows}(t_i - \Delta t_i) \right] \Delta t_i
\]

\[+ \sum \text{Discrete Additions} - \sum \text{Discrete Withdrawals} \]

Discrete Additions and Withdrawals are specified by checking the Additions and/or Withdrawals boxes (in the Discrete Additions & Withdrawal Requests section) and by specifying a link to a discrete change signal with an Add instruction:

Instantaneously Replacing the Current Quantity in a Pool

Read more: Basic Concepts of Discrete Event Modeling (page 352); Modeling Discrete Changes to a Pool (page 388).

In some cases, you may want to instantaneously replace the current value (the Quantity) of a Pool with a specified value.

This can be accomplished by checking the Additions field and specifying a link to a discrete change signal with a Replace instruction:

Read more: Basic Concepts of Discrete Event Modeling (page 352); Modeling Discrete Changes to a Pool (page 388).
Note: If you have specified an **Upper** or **Lower Bound**, and choose to Replace the current value of a Pool, the replacement value must be within the bounds (or GoldSim will display a fatal error message).

**Browser View of a Pool Element**

The browser view of a Pool element can have a large number of inputs and outputs (depending on which boxes are checked in the dialog, and how many **Inflows** and **Outflow Requests** you have). If the **Upper** and **Lower Bound** boxes are checked, the browser view looks like this:

*The names for the individual Outflows are user-defined.*

Note: Element inputs and outputs are only shown in the browser if you choose to **Show Element Subitems** (accessed via the browser context menu by right-clicking in the browser).

**Using Basic Function Elements**

There are nine basic function elements that are used to manipulate data in a GoldSim model:

- Expression elements;
- Extrema elements;
- Selector elements;
- Splitter elements;
- Allocator elements;
- Sum elements; and
- Logical elements (And, Or, Not).
These elements are described in detail in the topics listed below.

Expression elements are probably the most common function element that you will use in your models.

The properties dialog for an Expression element looks like this:

The properties dialog for an Expression is very simple, and is almost identical to that of the Data element. Other than the **ID** and **Description**, there is only a single input field. Typically, you will enter a mathematical expression into this field. (You could, of course, just enter a number. In such a case, however, it would be clearer to someone viewing your model if you used a Data element for this).

Note that the input field for an Expression (the "Equation") is always expanded to several lines to better accommodate and display large equations.

Expression elements have a single output, which can be a value or a condition, and can be specified as a scalar, a vector or a matrix. You can specify these attributes by pressing the **Type...** button. By default, a new Expression element is a scalar, dimensionless value.

**Read more**: [Using Vectors and Matrices](#) (page 782).

The browser view of an Expression element shows a single input (referred to as the Definition), and a single output:

---

**Note**: Element inputs and outputs are only shown in the browser if you choose to **Show element subitems** (accessed via the browser context menu by right-clicking in the browser).

An example model which uses Expression elements (Expression.gsm) can be found in the General Examples folder in your GoldSim directory (accessed by selecting **File | Open Example...** from the main menu).
Extrema Elements

An Extrema element computes the maximum value (peak) or minimum value (valley) achieved by its input during a simulation. For example, in the figure shown below, the Peak and Valley curves are the outputs of Extrema elements with X as an input:

Extrema elements are very useful for tracking things like the peak contaminant concentration in a river, the peak number of customers in a check-out line, or the lowest water level reached in a water supply reservoir.

The properties dialog for an Extrema element looks like this:

An Extrema element has a single primary input (the value you wish to track). This input must have the same attributes (order and dimension) as the Extrema element, and can be a link or an expression. The Type determines whether Peak (maxima) or Valley (minima) values are tracked (as illustrated in the example
shown above). Note that the default symbol for the Extrema changes depending on whether you are tracking a Peak or a Valley:

Peak

Valley

Extrema elements have two outputs, one of which is optional. The primary output is the actual peak or valley. It is always a value, but can be specified as a scalar, a vector or a matrix. You can specify these attributes by pressing the Type… button. By default, a new Extrema element is a scalar, dimensionless value. You can also use Extrema elements to operate on and/or create vectors and matrices.

Read more: Using Vectors and Matrices (page 782).

The secondary output (which is optional) is the time of the greatest peak or valley. This output is only created if Record Time of Occurrence of Peak/Valley is checked. If it is checked, this output represents the elapsed time at which the peak (or valley) last occurred. The output is always a value with dimensions of time. It has the same order (scalar, vector, matrix) as the primary output.

Extrema elements have two options which affect how the primary output of the Extrema is computed:

- **Use Absolute Value of the Input**: If this is checked, GoldSim uses the absolute value of the input to determine the Peak or Valley, but the primary output retains the sign of the input (i.e., it could be negative). Hence, if you were computing the Peak, the output would represent the value of the Input which has had the highest absolute value since the beginning of the simulation (or since the element has been reset – see below). If you were computing the Valley, the output would represent the value of the Input which has had the lowest absolute value since the beginning of the simulation (or since the element has been reset – see below).

- **Reset when Triggered**: If this is checked, the Reset… button becomes available, which provides access to a standard GoldSim Trigger dialog.

Read more: Understanding Event Triggering (page 355).

When an Extrema is reset by a trigger, the Peak/Valley calculation is restarted on the next update (ignoring everything that has occurred previously, as if it was the start of the simulation again).

The browser view of an Extrema element shows a single input and two outputs (if you choose to save the time of the peak/valley). Additional inputs are also present if the element is being reset:

```
Browser
```

“TofExtreme” is the (optional) time of extreme output.
Note: Element inputs and outputs are only shown in the browser if you choose to Show Element Subitems (accessed via the browser context menu by right-clicking in the browser).

A simple example model which uses an Extrema element (Extrema.gsm) can be found in the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

In many simulations, you will want to use if..then logic in your model to specify how a particular parameter changes as a function of some state of the system (e.g., time).

For example, you might want to use if…then logic to define how a particular variable changes with time: if(ETime < 10 yr, 5, 6). Simple relationships such as these can readily be entered into an Expression element. In some situations, however, you will need to define complex, nested if..then logic to represent your system. Consider, for example, the following complex nested if..then statement:

if  A <10  
then X = 5  
else if A < 20  
then X = 6  
else if A < 40 and B > 2  
then X = 7  
else X=8

This would be represented within an Expression element using GoldSim's built-in if..then function and relational operators as follows:

\[ \text{if}(A < 10, 5, \text{if}(a < 20, 6, \text{if}(a < 40 \text{ and } b > 2, 7, 8))) \]

Needless to say, a nested if...then statement such as this is cumbersome to enter (and difficult to interpret once it has been entered). The Selector element was specifically designed to facilitate the representation of such logic.

The properties dialog for a Selector element looks like this:
Selector elements have a single output, which can be a value or a condition, and can be specified as a scalar, a vector or a matrix. You can specify these attributes by pressing the Type… button. By default, a new Selector element is a scalar, dimensionless value. You can also use Selector elements to operate on and/or create vectors and matrices.

Read more: Using Vectors and Matrices (page 782).

Each switch (row) in the Selector dialog represents a single if…then statement and has two parts: the condition (the first column), and the result if the condition is true (the second column). The if statements are evaluated in order, and the Selector takes on the value corresponding to the first true statement that is encountered. All Selectors have a final else statement, which represents the result if all the conditions are false.

A new Selector has a single switch. A Selector with a single switch has the same functionality as a simple if..then statement: If(A then B else C).

You can add switches using the Add Switch button. New switches are added below the switch which is currently selected. If the final else is selected, however, the switch will be inserted above it. If the first switch is selected, you will be prompted for whether you want to insert the new switch above or below the selected switch.

To delete a switch, select it (it will turn yellow) and press Delete Switch. Note that the last result (the else) cannot be deleted.

You can use the Move Up and Move Down buttons to change the order of the switches (and the order is important!).

The example discussed above would be represented in a Selector as follows:
This is clearly easier to enter and read than using nested \textit{if..then} functions.

\textbf{Note:} The dialog for a Selector is wider than other elements to allow long expressions to be easily displayed in the two columns. Note, however, that if an expression does not completely fit into one of the fields, like other input fields in GoldSim, the field will expand and the expression will wrap when you click into it. In addition, the Selector dialog is resizable. This allows you to stretch the dialog so as to see all of the expressions that you may have entered into the dialog at once.

Both the condition column and the result column can be entered as links or expressions. The first column (the condition) must either be:

- a link which is a scalar condition;
- a conditional expression; or
- the words "True" or "False".

\textit{Read more:} Creating Conditional Expressions (page 130).

The entries in the result column must have the same data type, order and dimensions as the element's output.

As shown below, the browser view of a Selector element shows a single output, and the following inputs: an \textit{Else} (the result if all conditions are false), and two inputs for every switch (\textit{If}_n and \textit{Then}_n, where \textit{n} is the switch number):
Using Basic Function Elements

Note: Element inputs and outputs are only shown in the browser if you choose to Show Element Subitems (accessed via the browser context menu by right-clicking in the browser).

A simple example model which uses a Selector element (Selector.gsm) can be found in the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

**Splitter Elements**

Splitter elements split an incoming signal between a number of outputs based on specified fractions or amounts. Typically, the signal will be a flow of material (e.g., water), but it could also be a discrete transaction.

The properties dialog for a Splitter element looks like this:

The incoming signal to a Splitter element (the **Amount**) must be a scalar. The **Type** drop-list allows you to specify whether the incoming signal is a Value or a Discrete Change Signal.

If it is a Discrete Change Signal, the icon for the Splitter indicates this by changing its color (from blue to red):
A Splitter can accept multiple discrete changes. This is indicated in the input field by separating the individual discrete change signals by semi-colons (e.g., Change1; Change2; Change3). You can also specify the multiple discrete changes using the Multiple Discrete Signals button, which displays a table listing the multiple discrete change signals.

**Multiple Discrete Signals button**

**Read more:** Propagating Discrete Signals Between Elements (page 353).

The outputs of the Splitter element are specified in the "Outputs" section of the dialog. By default, a Splitter element initially has two outputs (named Output1 and Balance). You can add additional outputs by pressing the Add button (the plus sign) below the list. Outputs can be deleted using the Delete button (the X), and moved up and down in the list using the two buttons to the right of the Delete button.

The names of the outputs can be changed by editing the items in the **Name** column.

By default, output amounts are entered as fractions of the input amount. However, you can also specify the output amounts directly (as values). The drop-list directly above the list of outputs specifies how outputs are entered:

**If output amounts are defined as fractions of the input amount, then:**

- For each output, you specify the **Fraction** of the incoming **Amount** that is assigned to that output. The Fractions can be specified as constants or expressions and/or links, and may change with time.

- If the **Sum of Fractions must equal one** checkbox is checked (the default), then the total amount of the input will be conserved, and:
  - Each Fraction must always be greater than or equal to zero.
  - You cannot specify the Fraction for the last output. GoldSim automatically assigns the remainder to this output such that the fractions add to one.
  - If the specified Fractions exceed one at any time during a simulation, GoldSim issues a fatal error.

- If the **Sum of Fractions must equal one** checkbox is cleared, then:
  - There are no constraints on the values for the Fractions (they can be negative or can exceed one).
  - You must specify the Fraction for the last output (GoldSim does not automatically assign the remainder to this output such that the fractions add to one).
The specified Fractions do not need to sum to one.

If output amounts are defined directly as amounts, then:

- For each output, you specify the Amount that is assigned to that output. The output Amounts can be specified as constants or expressions and/or links, and may change with time.

- If the Sum of Outputs must equal input amount checkbox is checked (the default), then the total amount of the input will be conserved, and:
  - You cannot specify the output Amount for the last output. GoldSim automatically assigns the remainder to this output such that the outputs add to input amount.
  - If the specified output Amounts exceed the input amount at any time during a simulation, GoldSim issues a fatal error.

- If the Sum of Outputs must equal input amount checkbox is cleared, then:
  - There are no constraints on the values for the Amounts (they can be negative or can exceed the input Amount).
  - You must specify the output Amount for the last output (GoldSim does not automatically assign the remainder to this output such that the output Amounts add to the input amount).
  - The specified output Amounts do not need to sum to the input amount.

Under some circumstances, you may want to specify the outputs as some function of the input amount. GoldSim facilitates this by providing a locally available property called "Amount". This represents the current value in the Amount field. You would reference this value as "~Amount":

This can be particularly useful for routing discrete change signals based on their value.

Read more: Using Splitter Elements to Route Discrete Changes Based on Their Value (page 397).

Note: “Amount” is an example of a locally available property. As such, it only has meaning inside the Fraction field, and cannot be referenced anywhere else.

Read more: Understanding Locally Available Properties (page 808).

As shown below in the browser view of a Splitter element, a Splitter has an output for each item in the Outputs list:
Note: Element inputs and outputs are only shown in the browser if you choose to Show Element Subitems (accessed via the browser context menu by right-clicking in the browser).

The outputs of a Splitter can be either values or discrete change signals (depending on how the Type field was defined).

Note: If a Splitter outputs discrete change signals, the outputs preserve the Instruction of the input signal.

Note: If a Splitter outputs discrete change signals, signals with a zero value and an “Add” instruction are not propagated as outputs.

An example model which uses a Splitter element (Splitter.gsm) can be found in the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

Allocate Elements

Allocator elements allocate an incoming signal to a number of outputs according to a specified set of demands and priorities. Typically, the signal will be a flow of material (e.g., water), but it could also be a discrete transaction.

The properties dialog for an Allocator element looks like this:
The incoming signal to an Allocator element (the Amount) must be a scalar. The Type drop-list allows you to specify whether the incoming signal is a Value or a Discrete Change Signal.

If it is a Discrete Change Signal, the icon for the Allocator indicates this by changing its color (from blue to red):

An Allocator can accept multiple discrete changes. This is indicated in the input field by separating the individual discrete change signals by semi-colons (e.g., Change1; Change2; Change3). You can also specify the multiple discrete changes using the Multiple Discrete Signals button, which displays a table listing the multiple discrete change signals.

Read more: Propagating Discrete Signals Between Elements (page 353).

The outputs of the Allocator element are specified in the "Outputs" section of the dialog. By default, an Allocator element initially has one output (named Output1). You can add additional outputs by pressing the Add button (the plus sign) below the list. Outputs can be deleted using the Delete button (the X), and moved up and down in the list using the two buttons to the right of the Delete button.

The names of the outputs can be changed by editing the items in the Name column.

For each output, you specify the Demand for that output. The Demand has the same type and dimensions as the incoming Amount. Demands can be specified as constants or expressions and/or links, and may change with time. Each output receives its requested Demand according to the specified Priority. Outputs with lower valued Priorities are met first (i.e., Priority 1 is met before Priority 2).
To illustrate this, consider the following example:

<table>
<thead>
<tr>
<th>Priority</th>
<th>Name</th>
<th>Demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Output1</td>
<td>75 m³/day</td>
</tr>
<tr>
<td>2</td>
<td>Output2</td>
<td>20 m³/day</td>
</tr>
<tr>
<td>3</td>
<td>Output3</td>
<td>10 m³/day</td>
</tr>
<tr>
<td>4</td>
<td>Output4</td>
<td>5 m³/day</td>
</tr>
</tbody>
</table>

By default, each output has a fixed (and unique) Priority (which is an integer). The Priority of the outputs always decrease downward (with the first item having a Priority of 1). Priority 1 therefore represents the highest priority demand, 2 the next highest, and so on. New outputs are added below the output that is selected when pressing the Add button. You can move outputs up and down the Priority list using the Move Up and Move Down buttons (found directly below the list).

If you check the **Allow editing of Priorities** checkbox, the **Priority** column becomes editable. In this case, the Priorities can be specified as constants or expressions and/or links, and may change with time. They do not need to be integers and can be any real value (positive or negative). Outputs with lower valued Priorities are met first (e.g., Priority -2.4 is met before Priority 5.7).

If multiple outputs have the same Priority, you can specify the manner in which equal priorities are treated in the **For equal priorities** field. There are two options: 1) “share the input equally”; or 2) “share proportional to demands”.

In this case, Output1 would have a value of 75m³/day (its entire demand would be met), Output2 would have a value of 20m³/day (its entire demand would be met), Output3 would have a value of 5m³/day (only part of its demand would be met), Output4 would have a value of 0m³/day (none of its demand would be met), and the "Unused" output would have a value of 0m³/day.
• Output4 would receive its full demand (5 m$^3$/day); Output2 and Output3 would receive the same amount (5 m$^3$/day). This leaves 10 m$^3$/day to be allocated.

• Output2 and Output3 would then equally share the remaining 10 m$^3$/day (such that each would receive a total of 10 m$^3$/day). So Output3 and Output4 would meet their demands, and Output2 would not.

If “share proportional to demands” is selected, if there is not enough supply to meet all of the demands, it is shared proportionally according to the demand.

To illustrate this, consider the same example as above, with proportional sharing:

In this case, Output1 would still have a value of 75 m$^3$/day (its entire demand would be met), Output2, Output3 and Output4 would share the remainder (25 m$^3$/day) as follows:

• Output2 would receive $20/35 = 57.1\%$ of the remaining 25 m$^3$/day.
• Output3 would receive $10/35 = 28.6\%$ of the remaining 25 m$^3$/day.
• Output4 would receive $5/35 = 14.3\%$ of the remaining 25 m$^3$/day.

The table below summarizes the results for these two examples (note that the total supply is 100 m$^3$/day and the total demand is 110 m$^3$/day):

<table>
<thead>
<tr>
<th>Output</th>
<th>Priority</th>
<th>Demand (m$^3$/day)</th>
<th>Result with equal sharing (m$^3$/day)</th>
<th>Result with proportional sharing (m$^3$/day)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output1</td>
<td>1</td>
<td>75</td>
<td>75</td>
<td>75</td>
</tr>
<tr>
<td>Output2</td>
<td>2</td>
<td>20</td>
<td>10</td>
<td>14.3</td>
</tr>
<tr>
<td>Output3</td>
<td>2</td>
<td>10</td>
<td>10</td>
<td>7.1</td>
</tr>
<tr>
<td>Output4</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>3.6</td>
</tr>
</tbody>
</table>

Under some circumstances, you may want to define a Demand by referencing the total amount available, or the remainder after the previous Priority has been met. GoldSim facilitates this by providing two locally available properties:

• "Total". This represents the current value in the Amount field. You would reference this value as "~Total".
• "Remainder". This represents the amount remaining (of the original amount) after all senior priorities (i.e., Demands with lower-valued
Priorities) have been met. Hence, the Remainder takes on a different value in each row. You would reference this value as "~Remainder".

**Note:** In situations where several rows have equal priorities, the ~Remainder for a particular row is equal to the total amount available to the row divided by the number of remaining unmet equal priority rows. In this calculation, the equal priority rows are ordered not based on the order they where entered, but from smallest demand to largest demand.

**Note:** Total and Remainder are examples of locally available properties. As such, they only have meaning inside the **Demand** field, and cannot be referenced anywhere else.

**Read more:** [Understanding Locally Available Properties](page 808).

For example, if Output1 had a Demand of 100 m$^3$/day, and Output2 had a Demand that was half of what was remaining after Output1’s demand was met, you could define the Demands as follows:

<table>
<thead>
<tr>
<th>Priority</th>
<th>Name</th>
<th>Demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Output1</td>
<td>100 m$^3$/day</td>
</tr>
<tr>
<td>2</td>
<td>Output2</td>
<td>50% * ~Remainder</td>
</tr>
</tbody>
</table>

As shown below in the browser view of an Allocator element, there is an output for each item in the Outputs list. There is also an additional output (called "Unused") that represents the portion of the incoming signal that was unused (i.e., unallocated, or left over after the demands were met).

**Note:** Element inputs and outputs are only shown in the browser if you choose to **Show Element Subitems** (accessed via the browser context menu by right-clicking in the browser).

The outputs of an Allocator can be either values or discrete change signals (depending on how the **Type** field was defined).

**Note:** If an Allocator outputs discrete change signals, the outputs (including the Unused output) preserve the Instruction of the input signal.
Note: If an Allocator outputs discrete change signals, signals with a zero value and an “Add” Instruction are not propagated.

Several example models which use a Allocator element (Allocator.gsm) can be found in the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu). This file includes the use of an Allocator to simulate competing demands on a Reservoir.

Note: The algorithm described above is the same algorithm used to allocate the outflows from the Pool element. In fact, a Pool can be thought of as a combination of a Reservoir and an Allocator.

Read more: Pool Elements (page 252).

Sum Elements

Sum elements add values together. Although this can also be done using an Expression, a Sum element provides a somewhat more convenient and user-friendly way to create a sum if a large number of items are involved. More importantly, it provides a powerful visual indication of what operation it is carrying out, and this adds to the transparency of your model.

The properties dialog for a Sum element looks like this:

![Sum Properties Dialog]

Sum elements have a single output, which must be a value, and can be specified as a scalar, a vector or a matrix. You can specify whether the output is a scalar, vector or matrix by pressing the Type... button. By default, a new Sum element is a scalar, dimensionless value. You can also use Sum elements to operate on and/or create vectors and matrices.

Read more: Using Vectors and Matrices (page 782).

You add an item to the Sum by pressing the Add Input button. A browser tree showing all of the elements in the model will be displayed.
Note that this tree is organized by containment in the same manner as the main browser. To insert a link from this dialog, you select a specific output object (or an element having a primary output), and then press **OK**. If you press **Cancel**, GoldSim will insert a new blank (zero) item. In either case, the dialog will close and the item will be added to the list of inputs for the Sum:

<table>
<thead>
<tr>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input(s)</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>C</td>
</tr>
</tbody>
</table>

The inputs to the Sum must all have the same attributes (data type, order, dimensions) as the output. Note that the inputs can be expressions (in addition to simply links). You can edit any of the input fields after the input is added (and can insert links using the context menu for the input field).

As shown below, the browser view of a Sum element shows a single output, and an input for each item in the Sum:

![Browser view of a Sum element](image)

**Note**: Element inputs and outputs are only shown in the browser if you choose to **Show Element Subitems** (accessed via the browser context menu by right-clicking in the browser).

An example model which uses a Sum element (Sum.gsm) can be found in the General Examples folder in your GoldSim directory (accessed by selecting **File | Open Example...** from the main menu).

GoldSim provides a number of operators which allow you to create conditional expressions (e.g., A>B, A>B and C < D, A==B). Conditional expressions are powerful because they allow you to add conditional logic to your models.

**Read more**: [Creating Conditional Expressions](page 130).

If the conditional expression is simple, you can create the expression directly in the input field to an element. If, however, your conditional expression is very complex and/or it is important to make your conditional logic very transparent, GoldSim’s three logical elements (And, Or, Not) are very useful. You use these three simple elements to visually display conditional logic:

---

**Logical Elements**

---
And elements perform a logical And operation on a list of conditions. Their output is True if and only if all of their inputs are True. Although this can also be done using an Expression, an And element provides a somewhat more convenient and user-friendly way to do this, particularly if a large number of items are involved. More importantly, the element explicitly illustrates the operation being carried out, increasing the transparency of your model and allowing you to create easy-to-read conditional logic diagrams.

The properties dialog for an And element looks like this:

And elements have a single output, which is always a scalar condition. If all of the input conditions to an And element are True, the output is True; otherwise, the output is False.

You add an input to the element by pressing the Add Input button. A browser tree showing all of the elements in the model will be displayed.

Note that this tree is organized by containment in the same manner as the main browser. To insert a link from this dialog, you select a specific output object (or
an element having a primary output), and then press **OK**. If you press **Cancel**, GoldSim will insert a new blank (False) item. In either case, the dialog will close and the item will be added to the list of inputs for the element.

The inputs to the And element must all be conditions. Note, however, that the inputs can be conditional expressions (in addition to simply links). You can edit any of the input fields after the input is added (and can insert links using the context menu for the input field).

The browser view of an And element shows a single output, and an input for each input condition:

![Browser view of And element](image)

**Note:** Element inputs and outputs are only shown in the browser if you choose to **Show Element Subitems** (accessed via the browser context menu by right-clicking in the browser).

**Or Elements**

Or elements perform a logical Or operation on a list of conditions. Their output is True if any of their inputs are True. Although this can also be done using an Expression, an Or element provides a somewhat more convenient and user-friendly way to do this, particularly if a large number of items are involved. More importantly, the element explicitly illustrates the operation being carried out, increasing the transparency of your model and allowing you to create easy-to-read conditional logic diagrams.

The properties dialog for an Or element looks like this:

![Properties dialog for Or element](image)
Or elements have a single output, which is always a scalar condition. If all of the input conditions to an Or element are False, the output is False; otherwise, the output is True. You add an input to the element by pressing the **Add Input** button. A browser tree showing all of the elements in the model is displayed.

Note that this tree is organized by containment in the same manner as the main browser. To insert a link from this dialog, you select a specific output object (or an element having a primary output), and then press **OK**. If you press **Cancel**, GoldSim will insert a new blank (False) item. In either case, the dialog will close and the item will be added to the list of inputs for the element.

The inputs to the Or must all be conditions. Note, however, that the inputs can be conditional expressions (in addition to simply links). You can edit any of the input fields after the input is added (and can insert links using the context menu for the input field).

As shown below, the browser view of an Or element shows a single output, and an input for each input condition:

---

**Note:** Element inputs and outputs are only shown in the browser if you choose to **Show Element Subitems** (accessed via the browser context menu by right-clicking in the browser).

---

**Not Elements**

Not elements are very simple. They perform a logical Not operation on a single input argument. That is, the output of a Not element is simply the opposite of its conditional input. Although this can also be done using an Expression, a Not element explicitly illustrates the operation being carried out, increasing the transparency of your model and allowing you to create easy-to-read conditional logic diagrams.

The properties dialog for a Not element looks like this:

---

**GoldSim User’s Guide**  
**Chapter 4: Using the GoldSim Elements**  
• 293
The input to a Not element must be a scalar condition. It can, however, be an expression. Not elements have a single output, which is also a scalar condition. If the input to a Not is True, the output is False; if the input is False, the output is True.

Using Lookup Table Elements

Lookup Table elements are a type of input element that allow you to define your input in terms of a lookup table (or response surface). You define the table in terms of either one, two, or three independent variables.

For example, consider the following table:

<table>
<thead>
<tr>
<th>Length [m]</th>
<th>Mass [kg]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>6.4</td>
</tr>
<tr>
<td>20</td>
<td>7.7</td>
</tr>
<tr>
<td>40</td>
<td>8.6</td>
</tr>
<tr>
<td>60</td>
<td>9.9</td>
</tr>
<tr>
<td>100</td>
<td>15.6</td>
</tr>
</tbody>
</table>

This table defines the value of a dependent variable in terms of two independent variables. Hence, it is a two-dimensional lookup table. If the row variable was equal to 40 m, and the column variable was equal to 500 kg, the independent variable would take on a value of 11.3; if the row variable was equal to 60 m, and the column variable was equal to 1000 kg, the independent variable would take on a value of 17.5; and so on.

When you define a table like this in GoldSim, it determines the value of the dependent variable for any given values of the independent variables by interpolating between the data points supplied in the table. For example, if the row variable was equal to 30 m, and the column variable was equal to 500 kg, the independent variable would take on a value of 10.3 (assuming linear interpolation).

Look-up Table elements are unique in that you don’t reference them in the same manner that you reference the outputs of other elements. Instead, you reference them in the same manner that you would reference a built-in function (like sin or min). That is, once you define a table like the one above, you reference it in input expressions for other elements as if it were a custom function:

\[ \text{Equation} \]

\[ X_{\text{Table}}(15\text{m}, 150\text{kg}) \]

This expression instructs GoldSim to use the lookup table defined by the element \( X_{\text{Table}} \) and compute an output value based on a value for the row variable of 15m and a value for the column variable of 150kg.

An example model which uses a Lookup Table element (BasicTable.gsm) can be found in the General Examples/LookupTable folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

The manner in which you define and reference a Lookup Table is discussed in detail in the sections below.

Steps for Defining a Lookup Table

The dialog for a Lookup Table element looks like this:
The following steps are required to define a Lookup Table:

1. You first specify the **Table Dimensions**. To do so, you select one of three radio buttons: 1-D, 2-D or 3-D. This determines how many independent variables the Lookup Table has.

2. You specify the **Units** in which the data in the table will be specified for each independent variable. For a 1-D Table, you must specify the **Row Units**. For a 2-D table, you must specify the **Row** and **Column Units**. For the 3-D Table, you must specify the **Row**, **Column** and **Layer Units**. In all cases, you must also specify the **Result Units**. These are the units in which the data in the table will be specified for the dependent variable (i.e., the result or output of the table).

3. You can optionally specify a **Name** for each independent variable. This simply adds labels to the data table (in step 4 below) to make it easier for you to enter (and view) the data.

4. You specify data points which define your table. This consists of a dependent variable value for each of a number of specified combinations of the independent variables. You can specify this data table manually (by typing or pasting in the data), you can import the data from a text file, or you can link the data directly to an external data source (such as a spreadsheet file).

5. GoldSim determines the value of the dependent variable for any combination of values of the independent variables by interpolating between the data point supplied in the table. You must specify how you want to carry out the interpolation (e.g., linear, log, next lower), and how you want GoldSim to handle combinations of variables that are outside the range of data supplied in the table (e.g., should GoldSim extrapolate?).

Steps 4 and 5 are discussed in detail in the sections below.
To specify the data for a Lookup Table (i.e., define the entries for the actual table), you must first specify the **Table Dimensions**. To do so, you select one of three radio buttons: 1-D, 2-D or 3-D.

This determines how many independent variables the Lookup Table has.

After doing so, you specify the **Units** in which the data in the table will be specified for each independent variable. For a 1-D Table, you must specify the **Row Units**. For a 2-D table, you must specify the **Row** and **Column Units**. For the 3-D Table, you must specify the **Row**, **Column** and **Layer Units**.

You can optionally specify a **Name** for each independent variable. This simply adds labels to the data table (in step 4 below) to make it easier for you to enter (and view) the data:

In all cases, you must also specify the **Result Units**. These are the units in which the data in the table will be specified for the dependent variable (i.e., the result or output of the table).

Once you have done this, you can specify the data table manually (by typing or pasting in the data) or you can import the data from a text file.

The manner in which this is done for each type of table is discussed in detail in the sections below.

**Note:** In addition to defining the data table manually or importing from a text file, you can also link the data directly to an external data source (such as a spreadsheet file).

**Specifying Data in a 1-D Lookup Table Manually**

You specify the data in a 1-D Lookup Table element manually by pressing the **Edit Data...** button.

The following dialog will be displayed:

---

**Read more:** [Linking a Lookup Table to an External Data Source](page 305).

---
In this table, the Result Units are kg and the Row Units are m. Note also that the independent variable has been assigned a Name of “Length”.

By default, the table is created with a single row of zeros.

You add rows to the table using the Add Row(s) button. If you hold down the Ctrl key when you press this button, you will be prompted for the number of rows to add (otherwise, a single row will be added). To delete one or more rows, select them and press the Remove Row(s) button.

The first column of the table is where you enter the values for the independent (row) variable; the second column is where you enter the values for the dependent variable (the result).

The fields in the table only accept numbers (they do not accept links). You should not append units to the numbers. The assumed units for the values that you enter are the units specified for the two variables on the main Lookup Table dialog. These units are displayed in the column headers.

You can paste data from a spreadsheet, Word document or comma delimited text file into a 1-D table. To paste data from the Windows clipboard into a table, simply click once in the cell representing the upper left-hand corner of the region of the table into which you wish to paste the data, and press Ctrl+V.

When you paste data into a table, GoldSim will overwrite any data existing in the target region, and if necessary, will automatically expand the size of the table (i.e., add rows) to accommodate all of the data being pasted.

In addition to entering the data by hand, or pasting from another application, you can also import data directly from a specially formatted text file, a spreadsheet or a database. This is discussed further below.

The values of the Independent Variable must increase monotonically as you move downward in the table. When you press OK to close the dialog, GoldSim automatically sorts the rows into the correct order, and also deletes all empty rows.
Using Lookup Table Elements

**Note:** If, after defining a 1-D Table, you change the units for one of the variables without changing the dimensions (e.g., from feet to meters), GoldSim will automatically convert the existing data in the table to the new units. For example, if one of the entries for a dependent variable was 1 ft, and you changed the units for the dependent variable to meters, GoldSim would automatically convert the entry to 0.3048 meters. If the units are changed in such a way that the dimensions for the variable are different, the numbers will not be changed.

**Note:** If you are defining a 1-D Table, and the Row Variable represents the simulation time (i.e., Time or Etime), instead of using a Lookup Table element, you should use a Time Series element. This element is essentially a one-dimensional lookup table that is specifically designed to represent Time Series data (i.e., 1-D lookup tables in which the independent variable is time).

**Note:** When a Lookup Table is linked to a spreadsheet or database, you cannot edit the data once it has been imported; you can only view it. If you wish to edit the data, you must first remove the link to the spreadsheet or database (by changing **Data Source** to “None”).

**Importing 1-D Lookup Table Data from a Text File**

Read more: [Using Time Series Elements](page 186).

GoldSim allows you to directly import a text file representing a 1-D Lookup Table. The format for this text file is as follows:

- number of dimensions (in this case, 1)
- the number of rows
- row value 1, row value 2, …, row value n
- dependent value 1, dependent value 2, …, dependent value n

For example, the following table:
Could be defined using the following text file:

! Example File
1
5
1, 2, 3, 4, 5
10, 20, 30, 40, 50

The data in the file can be comma delimited or space delimited, and lines beginning with "!" are ignored.

To import the text file press the **Import Table**… button in the Edit 1-D Table dialog. You will be prompted for the name of the text file.

---

**Note:** In addition to importing from a text file into a Lookup Table directly from the Edit dialog, you can also link a text file to a Lookup Table, so that it is imported automatically when the file changes.

---

**Specifying Data in a 2-D Lookup Table Manually**

You specify the data in a 2-D Lookup Table element manually by pressing the **Edit Data**… button, which will display a dialog for defining the data in the table.

The following dialog will be displayed:
Using Lookup Table Elements

In this table, the Result Units are kg, the Row Units are m, and the Column Units are day. Note also that the row variable has been assigned a Name of “Length” and the column variable has been assigned a Name of “Time”.

By default, the table is created with a single column and a single row, and all the data values set to zero.

Within this dialog, you add rows and columns to the table using the Add Row(s) and Add Column(s) buttons. If you hold down the Ctrl key when you press these buttons, you will be prompted for the number of rows or columns to add (otherwise, a single row or column will be added). To delete one or more rows or columns, select them and press the Remove Row(s) or Remove Col(s) button.

The first column of the table is where you enter the values for the independent (row) variable; the first row of the table is where you enter the values for the second independent (column) variable, and the remainder of the table is where you enter the values for the dependent variable (the result).

The fields in the table only accept numbers (they do not accept links). You should not append units to the numbers. The assumed units for the values that you enter are the units specified for the three variables on the main Lookup Table dialog. These units are displayed in the row and column headers (for the Row and Column Variables) and at the top of the dialog (for the Dependent Variable).

You can paste data from a spreadsheet, Word document or comma delimited text file into a 2-D table. To paste data from the Windows clipboard into a table, simply click once in the cell representing the upper left-hand corner of the region of the table into which you wish to paste the data, and press Ctrl+V.

When you paste data into a table, GoldSim will overwrite any data existing in the target region, and if necessary, will automatically expand the size of the table (i.e., add rows and columns) to accommodate all of the data being pasted.

In addition to entering the data by hand, or pasting from another application, you can also import data directly from a specially formatted text file. This is discussed further below.
The values of the Row and Column Variables must increase monotonically as you move downward or across in the table. When you press OK to close the dialog, GoldSim automatically sorts the rows and columns into the correct order, and also deletes all empty rows and columns.

**Note:** If, after defining a 2-D Table, you change the units for one of the variables without changing the dimensions (e.g., from feet to meters), GoldSim will automatically convert the existing data in the table to the new units. For example, if one of the entries for a dependent variable was 1 ft, and you changed the units for the dependent variable to meters, GoldSim would automatically convert the entry to 0.3048 meters. If the units are changed in such a way that the dimensions for the variable are different, the numbers will not be changed.

**Note:** When a Lookup Table is linked to a spreadsheet or database, you cannot edit the data once it has been imported; you can only view it. If you wish to edit the data, you must first remove the link to the spreadsheet or database (by changing **Data Source** to “None”).

---

**Importing 2-D Lookup Table Data from a Text File**

GoldSim allows you to directly import a text file representing a 2-D Lookup Table. The format for this text file is as follows:

- number of dimensions (in this case, 2)
- the number of columns, the number of rows
- column value 1, column value 2, …, column value n
- row value1, row value 2, …, row value n
- dependent(row 1, column 1), …, dependent(row 1, column n)
- dependent(row 2, column 1), …, dependent(row 2, column n)
- …
- dependent(row n, column 1), …, dependent(row n, column n)

For example, the following table:
could be defined using the following text file:

```
! Example File
2
3, 4
1, 3, 9
1, 2, 4, 8
1, 1.5, 2.5
1.5, 2, 4
2.5, 3, 5
5, 6, 10
```

The data in the file can be comma delimited or space delimited, and lines beginning with "!" are ignored.

To import the text file press the **Import Table** button in the Edit 2-D Table dialog. You will be prompted for the name of the text file.

---

**Note:** In addition to importing from a text file into a Lookup Table directly from the Edit dialog, you can also link a text file to a Lookup Table, so that it is imported automatically when the file changes.

**Read more:** [Linking a Lookup Table to a Text File](#) (page 310).

**Specifying Data in a 3-D Lookup Table Manually**

You specify the data in a 3-D Lookup Table element manually by pressing the **Edit Data** button, which will display a dialog for defining the data in the table.
A 3-D table can be thought of as a 2-D table with multiple layers or slices. The Layer Variable defines the layer or "slice" of the look-up table, the Row Variable defines the rows, and the Column variable defines the columns.

The following dialog will be displayed:

In this table, the Result Units are kg, the Row Units are m, the Column Units are day and the Layer Units are m³. Note also that the row variable has been assigned a Name of "Length", the column variable has been assigned a Name of "Time" and the layer variable has been assigned a Name of "Volume".

By default, the table is created with a single column, a single row and a single layer, and all the data values set to zero.

Within this dialog, you view one 2-D layer or slice of the table at a time. Each 2-D layer is associated with a particular value of the Layer Variable. When you first create a 3-D Table, there is by default, a single layer (a single value for the Layer Variable), and it is equal to 0. You can edit the value for a Layer Variable by pressing the Edit Layer… button, which will display the following dialog:

You can only enter a number into this field and should not append units. The assumed units for the values that you enter are the units specified for the Layer Variables on the main Lookup Table dialog. These units are displayed just above the Layer Variable edit field.

You can add additional layers by pressing the Add Layer button (which will display the same dialog as shown above). The Delete Layer button deletes the current layer. You can edit the value of an existing layer variable by pressing the Edit Layer… button.
You add rows and columns to the table using the Add Row(s) and Add Column(s) buttons. If you hold down the Ctrl key when you press these buttons, you will be prompted for the number of rows or columns to add (otherwise, a single row or column will be added). To delete one or more rows or columns, select them and press the Remove Row(s) or Remove Col(s) button. Note that all Layers have the same rows and columns.

Within this dialog, the first column of the table is where you enter the values for the independent (row) variable; the first row of the table is where you enter the values for the second independent (column) variable, and the remainder of the table is where you enter the values for the dependent variable (the result).

The fields in the table only accept numbers (they do not accept links). You should not append units to the numbers. The assumed units for the values that you enter are the units specified for the three variables on the main Lookup Table dialog. These units are displayed in the row and column headers (for the Row and Column Variables) and at the top of the dialog (for the Dependent Variable).

You can paste data from a spreadsheet, Word document or comma delimited text file into a Layer of a 3-D table. To paste data from the Windows clipboard into a table, simply click once in the cell representing the upper left-hand corner of the region of the table into which you wish to paste the data, and press Ctrl+V.

When you paste data into a table, GoldSim will overwrite any data existing in the target region, and if necessary, will automatically expand the size of the table (i.e., add rows and columns) to accommodate all of the data being pasted.

In addition to entering the data by hand, or pasting from another application, you can also import data directly from a specially formatted text file. This is discussed further below.

The values of the Row, Column and Layer Variables must increase monotonically as you move downward or across in the table. When you press OK to close the dialog, GoldSim automatically sorts the rows and columns into the correct order, and also deletes all empty rows and columns.

Note: If, after defining a 3-D Table, you change the units for one of the variables without changing the dimensions (e.g., from feet to meters), GoldSim will automatically convert the existing data in the table to the new units. For example, if one of the entries for a dependent variable was 1 ft, and you changed the units for the dependent variable to meters, GoldSim would automatically convert the entry to 0.3048 meters. If the units are changed in such a way that the dimensions for the variable are different, the numbers will not be changed.

Note: When a Lookup Table is linked to a spreadsheet or database, you cannot edit the data once it has been imported; you can only view it. If you wish to edit the data, you must first remove the link to the spreadsheet or database (by changing Data Source to “None”).

Importing 3-D Lookup Table Data from a Text File

GoldSim allows you to directly import a text file representing a 3-D Lookup Table. The format for this text file is as follows:

- number of dimensions (must be 3)
• the number of columns, the number of rows, number of layers
• column value 1, column value 2, …, column value n
• row value 1, row value 2, …, row value n
• layer value 1, layer value 2, …, layer value n
• dependent(row 1, column 1, layer 1), …, dependent(row 1, column n, layer 1)
• dependent(row 2, column 1, layer 1), …, dependent(row 2, column n, layer 1)
• …
• dependent(row n, column 1, layer 1), …, dependent(row n, column n, layer 1)
• .
• .
• dependent(row 1, column 1, layer n), …, dependent(row 1, column n, layer n)
• dependent(row 2, column 1, layer n), …, dependent(row 2, column n, layer n)
• …
• dependent(row n, column 1, layer n), …, dependent(row n, column n, layer n)

The data in the file can be comma delimited or space delimited, and lines beginning with "!" are ignored.

To import the text file press the Import Table… button in the Edit 3-D Table dialog. You will be prompted for the name of the text file.

You can also import a text file representing a two-dimensional look-up table into a layer. To do this, you define a 2-D table in a text file and press the Import Layer… button. You will be prompted for the file to import. The data from the table overwrites the information in the current layer.

Read more: Importing 2-D Lookup Table Data from a Text File (page 301).

Note: In addition to importing from a text file into a Lookup Table directly from the Edit dialog, you can also link a text file to a Lookup Table, so that it is imported automatically when the file changes.

Read more: Linking a Lookup Table to a Text File (page 310).

In addition to allowing you to specify the data for a Lookup Table manually (by typing or pasting in the data, or manually importing the data from a text file), GoldSim also allows you to link the data directly to an external data source.

GoldSim provides four options for doing this:

• You can link the table to a spreadsheet file;
• You can link the table to a text file;
• You can link the table to a database; and
• You can link the table to an external program (a DLL) which outputs the table directly to the Lookup Table element.

You link a Lookup Table to an external source by selecting a data source type from the Data Source field at the bottom of the Lookup Table dialog:

By default, this is set to “None”. To link the Lookup Table to a Data Source, you select a different option from this list box (MS-Excel, Extended GoldSim Database, External DLL or ASCII Text File).

When you do so, a new tab is added to the dialog that allows you to define the properties of the Data Source link.

The four options for linking to an external Data Source are described in detail below.

Read more: Linking a Lookup Table to a Spreadsheet (page 306); Linking a Lookup Table to a Text File (page 310); Linking a Lookup Table to a Database (page 311); Defining a Lookup Table Using an External Function (page 312).

You link a Lookup Table to a spreadsheet by selecting “MS-Excel” from the Data Source field at the bottom of the Lookup Table dialog:

When you do so, a new tab (Excel) is added to the dialog that allows you to define the properties of the spreadsheet link:
You must first enter the name of a Microsoft Excel spreadsheet file by pressing the Options >> button. This will provide options for either selecting an existing file, or creating (and then selecting) a new file.

**Note:** If you select a file in the same directory as (or a subdirectory below) your GoldSim .gsm file, GoldSim will subsequently display just a local path. If you select a file in a directory above your .gsm file, it will display the full path.

Once you have selected a file, you can subsequently use the Options >> button to select a different file. You can also use the Options>> button to open the selected file in Excel.

**Note:** GoldSim supports both .xlsx and .xls Excel files. However, if you have an older version of Excel (prior to Office 2007), you will need to install Microsoft’s Office Compatibility Pack in order to read .xlsx files. Excel 2007 and later support an extended worksheet size (1,048,576 rows by 16,384 columns) than earlier versions (65,536 rows by 256 columns). If you wish to import data from an extended worksheet range into GoldSim, you must use Excel 2007 or newer, and the file format must be .xlsx. Note that GoldSim does not officially support versions of Excel prior to Excel 2003.

**Warning:** You should never try to open a spreadsheet that is linked to GoldSim while a simulation is running. In addition, prior to running a GoldSim model that is linked to a spreadsheet, all referenced spreadsheets should be closed. If you try to interact with a spreadsheet that is being used by GoldSim during a simulation, all interaction with the spreadsheet is blocked by GoldSim. However, under some circumstances this may not be possible and could lead to errors. Hence, as a general rule, you should not use Excel while a GoldSim model that references Excel is running.

It is easiest to understand how to define the spreadsheet linkage by considering 1-D, 2-D and 3-D tables separately.

### Defining 1-D Tables

- You must specify the **Row Variable Starting Cell**. This represents the first value of the Row Variable. Row variable data must be contiguous and proceed down in a column.

- You can specify the cell (e.g., A2) directly by typing it into the **Starting Cell** field. In this case, you must also select the sheet from the **Starting Sheet** drop-list. Alternatively, you can press the **Location...** button, which opens the spreadsheet and provides a dialog allowing you to directly select the desired cell.

- Dependent data are assumed to be in the column immediately to the right of the Row Variable column.

- The import of data stops when a cell is encountered that is empty or contains non-numeric data.
Defining 2-D Tables

- You must specify the **Row Variable Starting Cell** and the **Column Variable Starting Cell**. These represent the first value of the Row Variable and Column Variable, respectively. Row variable data must be contiguous and proceed down in a column. Column variable data must be contiguous and proceed to the right across a column. The **Column Variable Starting Cell** must be above the **Row Variable Starting Cell** in the spreadsheet.

- You can specify a cell (e.g., B1) directly by typing it into the **Starting Cell** fields. In this case, you must also select the sheet from the **Starting Sheet** drop-list. Alternatively, you can press the **Location…** button adjacent to either field, which opens the spreadsheet and provides a dialog allowing you to directly select the desired cell.

- Dependent data for a particular Row and Column Variable is assumed to be at the intersection of these two variables in the sheet. For example, if the **Row Variable Starting Cell** was A2 and the **Column Variable Starting Cell** was in B1, the dependent variable for the first row and the first column would be imported from B2:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
</tr>
<tr>
<td>5</td>
<td>40</td>
<td>18</td>
<td>20</td>
<td>22</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If the **Row Variable Starting Cell** was specified as A2, the **Row Variable values** would be imported as 10, 20, 30, 40 and 50. If the **Column Variable Starting Cell** was specified as B1, the **Column Variable values** would be imported as 3, 6, 9, 12, and 15. B2 would be imported as representing the dependent variable when the **Row Variable was equal to 10 and the Column Variable was equal to 3. In this file A1 would never be used.**

- The import of data stops when a cell is encountered that is empty or contains non-numeric data.

Defining 3-D Tables

- You must specify the **Row Variable Starting Cell**, the **Column Variable Starting Cell** and the **Layer Variable Starting Cell**. These represent the first value of the Row Variable, Column Variable and Layer Variable, respectively. Row variable data must be contiguous and proceed down in a column. Column variable data must be contiguous and proceed to the right across a column. The **Column Variable Starting Cell** must be above the **Row Variable Starting Cell** in the spreadsheet. Layers are represented in separate sheets. The value for the Layer Variable for each sheet must always be in the same cell.

- You can specify a cell (e.g., B1) directly by typing it into the **Starting Cell** fields. Alternatively, you can press the **Location…** button adjacent to either field, which opens the spreadsheet and provides a dialog allowing you to directly select the desired cell.

- The **Starting Sheet** represents the location of the data for the first Layer Variable. Additional Layers are assumed to be to the right of the **Starting Sheet**.
The values of the Row and Column Variables must be the same in every Layer sheet. If they are not, GoldSim will not import the table.

Dependent data for a particular Row and Column Variable in a particular Layer sheet is assumed to be at the intersection of these two variables in the sheet. For example, if the **Row Variable Starting Cell** was A2 and the **Column Variable Starting Cell** was in the B1, the dependent variable for the first row and the first column would be imported from B2:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>6</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>16</td>
<td>10</td>
<td>20</td>
<td>22</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>21</td>
<td>23</td>
<td>27</td>
<td>26</td>
<td>33</td>
</tr>
</tbody>
</table>

*If the Row Variable Starting Cell was specified as A2, the Row Variable values would be imported as 10, 20, 30, 40, and 50. If the Column Variable Starting Cell was specified as B1, the Column Variable values would be imported as 3, 6, 9, 12, and 15. B2 would be imported as representing the dependent variable when the Row Variable was equal to 10 and the Column Variable was equal to 3. In this file, A1 represents the value of the Layer Variable for this sheet.*

The import of data stops when a cell is encountered that is empty or contains non-numeric data (where it expects to find a layer number).

**Note:** When data is imported from Excel into a Lookup Table, it is assumed that the values for the independent and dependent variables in Excel are in the same units as those specified in the Lookup Table dialog.

**Note:** When data is imported from Excel into a Lookup Table, it will stop the import and report an error if the values of the independent variables do not increase monotonically as you move downward/ across the table. That is, unlike the case when you enter the data directly, GoldSim will not automatically sort the entries.

GoldSim automatically imports data from the spreadsheet at the start of a simulation if either 1) any of the properties on the Excel tab have been modified; or 2) the Excel file itself has been changed since the last import. You can also import data manually at any time prior to running a simulation (e.g., so you can view it) by pressing the **Import Data** button in the Excel tab.

**Note:** When you link a Lookup Table to a spreadsheet, the **Edit Data** button changes to **View Data**. If you press this button to access the dialog displaying the data, you will note that they are no longer editable. Once you are linked to a spreadsheet in this way, you can not edit the data manually unless you change **Data Source** back to “None”. When you do so, GoldSim keeps the imported data and makes it editable.

If you want to ensure that you do not import data from the spreadsheet that has been inadvertently edited since you last imported the data, you can choose to “Lock onto” a spreadsheet file (by checking “Lock onto selected file” from the **Options** button). If you are locked onto a file, GoldSim will not allow the
Using Lookup Table Elements

simulation to run if the file has been modified in any way (the file is set to read only when it is locked onto). In order to run a simulation with a changed file, you must first remove the lock (by clearing “Lock onto selected file” from the Options>> button).

The example file (ImportTable.gsm) in the General Examples/LookupTable folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) illustrates how the Lookup Table data can be imported from a spreadsheet.

You link a Lookup Table to a text file (at runtime) by selecting “ASCII Text File” from the Data Source field at the bottom of the Lookup Table dialog:

When you do so, a new tab (Text File) is added to the dialog that allows you to define the name of the text file:

You must first enter the name of a text file by pressing the Options >> button.

Note: If you select a file in the same directory as (or a subdirectory below) your GoldSim .gsm file, GoldSim will subsequently display just a local path. If you select a file in a directory above your .gsm file, it will display the full path.

Once you have selected a file, you can subsequently use the Options >> button to select a different file. You can also use the Options>> button to open the selected file in your default text editor.

The format for the text file is identical to the format used when importing into a Lookup Table directly from the Editing dialog.
Read more: Importing 1-D Lookup Table Data from a Text File (page 298); Importing 2-D Lookup Table Data from a Text File (page 301); Importing 3-D Lookup Table Data from a Text File (page 304).

Note: When data is imported from a text file into a Lookup Table, it will stop the import and report an error if the values of the independent variables do not increase monotonically as you move downward/across the table. That is, unlike the case when you enter the data directly, GoldSim will not automatically sort the entries.

GoldSim automatically imports data from the text file at the start of a simulation if either 1) the filename has been changed; or 2) the file itself has been changed since the last import. You can also import data manually at any time prior to running a simulation (e.g., so you can view it) by pressing the Import Data button in the Text File tab.

Note: When you link a Lookup Table to a text file, the Edit Data… button changes to View Data…. If you press this button to access the dialog displaying the data, you will note that they are no longer editable. Once you are linked to a text file in this way, you can not edit the data manually unless you change Data Source back to “None”. When you do so, GoldSim keeps the imported data and makes it editable.

If you want to ensure that you do not import data from the text file that has been inadvertently edited since you last imported the data, you can choose to “Lock onto” a text file (by checking “Lock onto selected file” from the Options>> button). If you are locked onto a file, GoldSim will not allow the simulation to run if the file has been modified in any way (the file is set to read only when it is locked onto). In order to run a simulation with a changed file, you must first remove the lock (by clearing “Lock onto selected file” from the Options>> button).

You link a Lookup Table to a database by selecting “Extended GoldSim Database” from the Data Source field at the bottom of the Lookup Table dialog:

![Database Linking](image)

When you do so, a new tab (Database) is added to the dialog.

A Lookup Table is one of several types of elements that can be linked to a database. All elements that can import data from a database use a common set of structures and rules for doing so. Hence, in order to use this capability in GoldSim, you must be familiar with these rules and database structures.

An Extended GoldSim Database is one of several available database structures supported by GoldSim, and the only structure that supports Lookup Tables.

Note: Only 1-D and 2-D Lookup Tables can be linked to a database. GoldSim does not support linking a 3-D Lookup Table to a database.
Using Lookup Table Elements

Read more: Linking Elements to a Database (page 1031).

Note: When you link a Lookup Table to a database, the Edit Data... button changes to View Data... If you press this button to access the dialog displaying the data, you will note that they are no longer editable. Once you are linked to a database in this way, you cannot edit the data manually unless you change Data Source back to “None”. When you do so, GoldSim keeps the imported data and makes it editable.

Defining a Lookup Table Using an External Function

In some cases, you may wish an external program to directly generate the data that you can use to dynamically define a Lookup Table element. To facilitate this, GoldSim allows you to output the table from a DLL using an external element.

You link a Lookup Table to an external function by selecting “External DLL” from the Data Source field at the bottom of the Lookup Table dialog:

When you do so, a new tab (External DLL) is added to the dialog. You use this tab to specify the name of the External element output that defines the table.

In order to do so, you must first be familiar with creating DLLs and linking them to GoldSim using External elements.

Read more: Using an External Element to Define Lookup Tables (page 937).

Note: When a Lookup Table is linked to an external DLL, you can view the data once the model has been run. If you subsequently wish to edit the data, you can do so by first removing the link to the external DLL (by changing Data Source to “None”).

Controlling Interpolation and Extrapolation for a Lookup Table

By default, the interpolation carried out by GoldSim to determine the value of the Dependent Variable for a given value of the Independent Variable (i.e., the argument) is done linearly with respect to both variables. However, you can change the interpolation to be used for any or all of the variables. This is done via the Interpolation drop-lists for the independent variables and the Result Interpolation drop-list for the dependent variable:

For the independent variables, GoldSim provides the following interpolation options:

Linear: This is the default, and if selected, GoldSim will linearly interpolate between table values to determine the appropriate value.
**Next lower**: This option results in a “stair-step” function, with no interpolation at all. If this option is selected, given an input argument for the table, GoldSim uses the largest independent variable entry in the table that is less than or equal to the input argument. For example, if your independent variable entries were 10, 20 and 30, and the input argument was 17, GoldSim would assume an independent variable value of 10.

**Exact only**: If this option is selected, the Lookup table will only accept arguments that exactly match independent variable entries in the table. If there is not an exact match, rather than interpolating between points, GoldSim issues a fatal error.

The only options available for Result Interpolation are **Linear** and **Log**.

The general guidance below can be used to determine when to use Linear or Log interpolation for the Result variable:

- If you plot the Results against an independent variable, and it becomes approximately a straight line, select Linear for both the Result and the Independent Variable.
- If you plot the logarithm of the Results against an independent variable, and it becomes approximately a straight line, select Log for Result and Linear for the Independent Variable.

**Note**: If an independent variable appears to vary logarithmically, enter the logs of the independent variable when defining points in the Lookup Table (and use the log of the independent variable as an argument when referencing the table).

When you define a Lookup Table, you must also specify how GoldSim is to handle references to the table that are outside the bounds of the table. For example, if your Row Variable varied between 1 and 10, and you used the Lookup Table to evaluate a value for the Row Variable equal to 15, how should GoldSim handle this?

The **Handling of Data Outside Bounds** field specifies how GoldSim is to handle such an instance. GoldSim provides three options for 1-D Tables, and two options for 2-D and 3-D Tables:

- **Fatal Error**: This is the default. If you try to reference a point outside the bounds of the table, GoldSim will display a Fatal Error message when you try to run the simulation.
- **Do Not Extrapolate**: If you try to reference a point outside the bounds of the table, GoldSim will not extrapolate. Instead, it will compute the value for the dependent variable based on the values for the independent variables at the nearest portion of the defined region. That is, it assigns a value from the nearest “edge” of the table.
- **Extrapolate**: This option is only available for 1-D Tables. GoldSim extrapolates beyond the range of the specified data by using the two closest data pairs.

An example model which illustrates various interpolation and extrapolation options (TableInterpolation.gsm) can be found in the General Examples/LookupTable folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).
Referencing a Lookup Table

Lookup Table elements are unique in that you don’t reference them in the same manner that you reference the outputs of other elements. Instead, you reference them in the way that you would reference a built-in function (like sin or min). That is, once you define a table, you reference it in input expressions for other elements as if it were a custom function:

\[
\text{Equation} \\
\text{X\_Table}(15\text{m}, 150\text{kg})
\]

This is referred to as a *table function*. This expression instructs GoldSim to use the 2-D lookup table defined by the element X\_Table and compute an output value based on a value for the row variable of 15m and a value for the column variable of 150kg.

Of course, once you have defined the table, you can reference it at multiple locations (i.e., in the input fields for various elements) using different input arguments (different independent variables) in the same manner as you would use the built-in functions provided by GoldSim multiple times. For example, elsewhere in the model you could reference X\_Table(10ft, 135kg).

The argument(s) to a table function must have dimensions that are the same as those of the independent variables with which the Lookup Table was originally defined. Hence, in the example above, if you referenced the table as X\_Table(10 day, 150 kg), GoldSim would produce an error message, since based on the way that X\_Table was defined, the first argument (the Row Variable) must have dimensions of length.

The dimensions of the table function are determined by the Result Units specified when the Lookup Table was defined.

Table functions are always referenced as follows:

1-D Tables: \(Tablename(\text{Row Variable})\)

2-D Tables: \(Tablename(\text{Row Variable, Column Variable})\)

3-D Tables: \(Tablename(\text{Row Variable, Column Variable, Layer Variable})\)

The arguments to a table function do not need to be numbers; they can be links from other elements.

Table functions can also accept arrays as input arguments (and subsequently produce arrays as outputs). For example, if ABC was a two-dimensional lookup table, \(ABC(1,2)\) would return a scalar value. However, if \(X\) and \(Y\) were vectors (e.g., of “Days”), then \(ABC(X,Y)\) would return a vector of days. The items would be evaluated by the lookup table on a term-by-term basis.

*Read more:* [Using Vectors and Matrices](page 782).

When you create a Lookup Table, GoldSim automatically lists the table function in the Function menu that is accessible from the context menu within an input field:
There are two ways to add a Lookup Table to an expression:

1. You can use the context menu to insert a Table function (as shown above).
2. You can use the Insert Link cursor to reference the table directly:

   In either case, the table is inserted as a function (i.e., with parentheses for arguments, and commas if there are multiple arguments):

   **X_Table()**

In addition to referencing a Lookup Table as described above, GoldSim also provides some specialized ways to reference a 1-D table (i.e., to do an “inverse lookup” and to compute the integral or derivative of a 1-D Table). These options are discussed below.

In some cases, you may want to do an inverse lookup into a Lookup Table. For example, suppose that you had defined a Lookup Table in which the independent variable is the volume of water in a pond, and the dependent variable is the water elevation in the pond (hence, the values in the table are determined by the shape of the pond). You would then use the Table to return a value for the water elevation for a specified value of the volume. However, what if you wanted to know the corresponding volume for a specified value of the water elevation?

GoldSim supports this by allowing you to carry out a reverse lookup; that is, you can reference a 1-D table by sending it a value for the dependent variable, and the function will return the corresponding value for the independent variable.

To implement this, you must add a second argument to the table function:

   **Tablename(Row Variable, TBL_Inverse)**

Using the example described above, if you had a Lookup Table (named Elevation) in which the independent variable is the volume of water in a pond, and the dependent variable is the water elevation in the pond, then the following expression would return the elevation corresponding to a volume of 150 m³:
Using Lookup Table Elements

and the following expression would return the volume corresponding to an elevation of 10 m:

Several points should be noted:

- If the argument is outside of the range of the table, GoldSim issues a fatal error (regardless of the Handling of Data Outside Bounds field setting).
- If the table is non-monotonic, the function returns the first value of the row variable that corresponds to the argument.
- The TBL_Inverse argument can only be used with 1-D tables. It cannot be used by 2-D and 3-D tables.

An example model which illustrates the inverse function (TableFunctions.gsm) can be found in the General Examples/LookupTable folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

In some cases, you may want to compute the integral or derivative of the function defined by a Lookup Table.

To implement these, you can add a second argument to the table function:

\[ \text{Tablename}(\text{Row Variable}, \text{TBL_Derivative}) \] This returns the derivative of the function at the specified row value.

\[ \text{Tablename}(\text{Row Variable}, \text{TBL_Integral}) \] This returns the integral of the table function up to the specified row value.

\[ \text{Tablename} (\text{Result Variable}, \text{TBL_Inv_Integral}) \] This returns the row value corresponding to a specified integral for the table result (the result variable).

For example, if you had a Lookup Table (named Volume) in which the independent variable was a length and the dependent variable was a volume, then the following expression would return the derivative of the corresponding function (an area) when the independent variable was 5 m:

\[ \text{Volume}(5 \text{ m}, \text{TBL_Derivative}) \]

**Note:** These arguments can only be used with 1-D tables. They cannot be used by 2-D and 3-D tables.

The best way to illustrate how these functions are calculated is to consider a simple example. Consider the following 1-D Table (named “Volume), in which the independent variable has dimensions of length and the dependent variable has dimensions of volume:
Computing the Derivative

The plot below shows how GoldSim would compute Volume(X, TBL_Derivative) for different values of X:

<table>
<thead>
<tr>
<th>Depth [m]</th>
<th>Result [m3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>45</td>
</tr>
</tbody>
</table>

Several points should be noted:

- For TBL_Derivative, the dimensions of the input argument must be that of the Independent Variable. The dimensions of the output of the function are Dependent Variable dimensions / Independent Variable dimensions. In this example, since the Independent Variable was a length and the Dependent Variable was a volume, the result of the function has dimensions of an area.
- If the input is outside of the range of the table, the function returns the value at the edge of the table (regardless of the Handling of Data Outside Bounds field setting).
- The derivative function always returns a value of zero if the table’s Interpolation is set to “Next lower”.
- The derivative function returns an error if the Interpolation is set to “Exact only”.

Computing the Integral

The plot below shows how GoldSim would compute Volume(X, TBL_Integral) for different values of X:
Several points should be noted:

- For TBL_Integral, the dimensions of the input argument must be that of the Independent Variable. The dimensions of the output of the function are Dependent Variable dimensions * Dependent Variable dimensions. In this example, since the Independent Variable was a length and the Dependent Variable was a volume, the result of the function has dimensions of length$^4$.

- The integral function integrates from the start of the table up to the lesser of the given argument and the end of the table. If the argument is less than the first row value for the table, the function returns zero.

- If the input is outside of the range of the table, the function returns the value at the edge of the table (regardless of the Handling of Data Outside Bounds field setting).

- The integral function returns an error if the Interpolation option is set to “Exact only”.

**Computing the Inverse Integral**

The plot below shows how GoldSim would compute Volume(X, TBL_Inv_Integral) for different values of X:
Several points should be noted:

- For TBL_Inv_Integral, the dimensions of the input argument must be Dependent Variable dimensions * Independent Variable dimensions. In this example, since the Independent Variable was a length and the Dependent Variable was a volume, the dimensions of the input argument must be length^4. The dimensions of the output of the function are those of the Independent Variable.

- If the input (the integral) is outside of the range produced by the table’s inputs, the function returns the value at the edge of the table (regardless of the Handling of Data Outside Bounds field setting). For example, in the example provided above, a value of 15m (the last data point in the table) corresponds to an integral value of 395m^4. As a result, if the first argument to Table(X, Inv_Integral) was greater than 395m^4, the function would return 15m.

- If the integral of the table is non-monotonic, the inverse integral returns the lowest value whose integral matches the argument.

- The inverse integral function returns an error if the Interpolation option is set to “Exact only”.

One practical application of the use of these functions is when creating tables that describe how the surface area, volume and water elevation in a pond or reservoir are related. In these kinds of problems, you often would compute the volume, and then use a table to output the surface area and elevation corresponding to that volume. One way to do this in a consistent way using the table functions would be to define a table (named, say, Area_from_Elevation) that outputs the surface area as a function of the water elevation (i.e., the independent variable would be elevation and the dependent variable would be surface area). Given a volume (V), you could then use this table to compute the elevation and surface area as follows:

\[
\text{Elevation} = \text{Area_from_Elevation}(V, \text{TBL_Inv_Integral})
\]

\[
\text{Surface_Area} = \text{Area_from_Elevation(Elevation)}
\]

An example model which illustrates these functions (TableFunctions.gsm) can be found in the General Examples/LookupTable folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).
Building a Dynamic Lookup Table

Because all the values defining a Lookup Table must be constants, the definition of the table cannot change dynamically.

In some situations, however, you may need to change the definition of the table as a function of time. There are two ways to build dynamic lookup table logic:

- You can define multiple Lookup Table elements, and use if,then logic to switch between the different tables as a function of time; or
- You can use a vector to define a lookup table. GoldSim provides several array functions that allow you to interpolate into the entries in a vector. Since vector entries can change with time, this allows you to use this functionality to create a dynamic lookup table.

*Read more:* [Using a Vector as a Lookup Table](#) (page 798).

A Lookup Table is an unusual element in that it does not have any inputs that accept links from other elements (since the entries to the table must be numbers). As a result, when you view a Lookup Table in the graphics pane, you will note that it never has an input port.

*Read more:* [Understanding Input and Output Ports](#) (page 76).

In addition, the output of a Lookup Table itself is not an actual value; rather it is simply a definition that does not take on a value unless you add the appropriate arguments.

As a result, the browser view of a Lookup Table is unusual in that it does not show any inputs or outputs (even when you have chosen the option to display these in the browser by selecting *Show Element Subitems*):

![Browser View of a Lookup Table Element](image)

*Note:* If the table is defined using an external function, the Table does have an input port and a single input – the table definition.

*Read more:* [Defining a Lookup Table Using an External Function](#) (page 312).

Using Delay Elements

Delay elements simulate processes that delay continuous or discrete signals and flows. That is, their outputs lag their inputs. Therefore, like Stocks, their outputs are functions of prior values of their inputs. As a result, Delays impart inertia and memory to a system, and can internally generate dynamic behavior within a system. Without elements like Delays and Stocks, your models could respond to outside (exogenous) drivers, but could not generate any dynamics of their own internally.

Depending on the structure of your model, a Delay can lead to instability and oscillation, or it can act to filter out unwanted noise.

GoldSim provides four Delay elements:
• **Information Delays** are intended to represent processes such as delays in measuring or reporting variables (e.g., reporting the inventory in a warehouse, or snow pack levels), and the gradual adjustment of perceptions based on available information (e.g., sales forecasts).

• **Material Delays** are intended to represent delays in the physical movement (flow) of material through a system (letters though the mail system, parts on an assembly line, salmon in a river, water moving though a pipe). Material is conserved as it moves through a Material Delay. Information is not conserved in an Information Delay.

• **Event Delays** provide a mechanism for delaying a discrete event signal (i.e., a discrete signal that indicates that something, such as an accident or a bank deposit, has occurred). Among other things, Event Delays can be used to simulate queues.

• **Discrete Change Delays** provide a mechanism for delaying a discrete change signal (i.e., a discrete signal that can instantaneously change the value of a Stock element).

Information Delays and Materials Delays are used to delay continuous signals and flows (and are discussed in the sections below).

Event Delays and Discrete Change Delays are used for representing delays in discrete signals. Use of these two elements requires an understanding of how GoldSim simulates discrete events.

**Read more:** Chapter 5: Simulating Discrete Events (page 351).

Information Delay elements are intended to be used to simulate delays in measuring, reporting, and/or responding to information. Such delays exist because it invariably takes time to collect, assimilate and act on new information.

These delays often have a critical impact on the dynamic behavior of systems. For example, many systems that are observed to oscillate involve such information delays. Example applications include processes such as delays in measuring or reporting variables (e.g., reporting the inventory in a warehouse, or snow pack levels), and the gradual adjustment of perceptions based on available information (e.g., sales forecasts).

Reporting yesterday’s rainfall total today is an example of an information delay, with a delay time of one day. In the business world, a classic example of an information delay is the delay in perceiving a change in a variable (e.g., the sales rate). The figure below illustrates the information delay associated with a manager’s perception of a change in the sales rate:
In all of these examples, the reported or perceived value (i.e., the output of the Information Delay) is computed based on the historical values of its inputs.

Note that because Information Delays usually represent processes such as perception, measurement and reporting, by definition, these elements are typically used to simulate human actions. In particular, they are often used to represent the decision-making process of someone (or some group) in the system being simulated. Hence, models that simulate a purely physical process in which humans play no part rarely use Information Delay elements.

The properties dialog for an Information Delay element looks like this:

An Information Delay requires an Input Signal and a Delay Time. If desired, you can optionally define the degree of Dispersion. The single output of the Information Delay is the delayed signal, which has the same dimensions and order as the signal itself.
Using Delay Elements

The best way to understand how to use Information Delays is to examine the behavior of the element in a number of simple examples. A file containing these examples (InformationDelay.gsm) can be found in the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

Within the Information Delay properties dialog, the Display Units determine the dimensions of the Information Delay’s output. This output can only be specified as a value (it cannot be a condition), and can be specified as a scalar, a vector or a matrix. You can specify whether the Information Delay is a scalar, vector or matrix by pressing the Type... button. By default, the output of a new Information Delay element is a scalar, dimensionless value. You can, however, use Information Delay elements to operate on and create vectors and matrices. Read more: Using Vectors and Matrices (page 782).

The Input Signal and the Initial Value inputs to the Information Delay must have the same attributes (order and dimensions) as the output.

**Note:** The Initial Value must be a number or a link from a static variable (e.g., a constant Data element or a Stochastic).

The Delay Time must have dimensions of time and must be positive.

**Note:** The Delay Time for an Information Delay must be greater than or equal to the timestep. That is, GoldSim cannot delay a signal for a smaller time period than a timestep. If you enter a Delay Time which is less than a timestep, GoldSim internally treats the Delay Time as being equal to a timestep. Delay Times that are less than or equal to zero will result in a fatal error.

The Dispersion drop-list provides three choices: “None” (the default), “Erlang n”, and “Std. Deviation”.

If one of the latter two is selected, you must enter a value which quantifies the degree of dispersion to the right of this field. If “Erlang n” is selected, you must enter a dimensionless value greater than or equal to 1. If “Std. Deviation” is selected, you must enter a value with dimensions of time. The value must be greater than or equal to zero and less than or equal to the Delay Time.

If the Information Delay is specified as being an array (i.e., a vector or a matrix), you can specify whether the Delay Time and Dispersion are defined as scalars (with the same value being applied to all items of the array), or as arrays (with different values applied to each item of the array). This is done via the Use scalar delay time and dispersion checkbox. If this box is checked (the default), the Delay Time and Dispersion must be entered as scalars. If the box is cleared (which is only possible if the element is specified as an array), the Delay Time and Dispersion must be entered as arrays.

The simplest use of an Information Delay element is to represent the delay of a non-dispersed signal. To simulate such a system, the Dispersion must be set to “None”. In this case, the output at any time t is simply the input lagged by the Delay Time:

\[
\text{Output}(t) = \text{Input}(t - \text{Delay Time})
\]

An example of such a delay is any process where a variable is instantaneously measured (e.g., the inventory in a warehouse) but not reported or received by a
manager or decision-maker until after a Delay Time. In the example shown below, the variable changes as a step, but there is a 5 day delay in reporting the value:

In the somewhat more interesting example shown below, the variable varies continuously (as a sine curve with a period of 15 days), and there is a 5 day delay in reporting the value. The Initial Value of the output of the Delay is set to 10:

In many cases, an information signal is dispersed, “smoothed” or “smeared” such that the output represents a weighted average of previous values for the input signal. (With no dispersion, the weight for the input value at \( t - \text{Delay Time} \) is one, and all other previous inputs have a weight of zero).

To specify that the information is dispersed, you must select either “Erlang n” or “Std. Deviation” from the Dispersion drop-list. These are two alternative ways to quantify the degree of dispersion in the signal.

Read more: Mathematics of Information Delays (page 327).
If “Erlang n” is selected, you must enter a dimensionless value greater than or equal to 1. As n increases, the degree of dispersion decreases. As n goes to infinity, the dispersion goes to zero. The maximum amount of dispersion allowed is represented by n = 1.

If “Std. Deviation” is selected, you must enter a value with dimensions of time. The value must be greater than or equal to zero and less than or equal to the Delay Time. As the Std. Deviation decreases, the degree of dispersion decreases. When the Std. Deviation goes to zero, the dispersion goes to zero. The maximum amount of dispersion allowed is represented by Std. Deviation = Delay Time.

The Erlang n and the Std. Deviation are related by the following equation:

\[ n = \left( \frac{\text{Delay Time}}{\text{Std. Deviation}} \right)^2 \]

The figure below shows the response of an Information Delay to a step function for various values of n:

Note that n = 1 (the highest level of dispersion allowed) is a special case referred to as exponential smoothing. In this case, the calculation carried out by the Information Delay element to compute its output is equivalent to exponentially weighting the previous values (i.e., the most recent value has the highest weight, and the weights of older values decrease exponentially). Exponential smoothing is a commonly used forecasting model.

Read more: Simulating Forecasts Using Information Delays (page 328).

Information Delays always have a specified Initial Value. The Initial Value is assumed to have been the value of the input signal prior to the start of your simulation. That is, it is the initial output of the Information Delay element. The default is zero.

The ability to define an Initial Value is important since the actual processes that you will be simulating may not necessarily start when your simulation begins. That is, in many cases, the process probably will have been ongoing for some time prior to the time that you decide to simulate it.
As a result, whatever your Information Delay actually represents (e.g., a perception, a forecast) will have some real value at the start of the simulation, and this value is not necessarily zero (the default). Therefore, you should take care to properly define an Initial Value for your Information Delay.

In some cases, the Delay Time for your Information Delay may change as a function of time. For example, a company may recognize and respond to improving economic conditions faster than deteriorating conditions. The figure below shows a situation in which the time to perceive a change in the system is faster when the current perceived state is less than the actual state (i.e., the state of the system has improved) and slower when the current perceived state is greater than the actual state (i.e., the state of the system has deteriorated).

To better understand how an Information Delay behaves when the Delay Time changes, it is worthwhile to consider a simple example. In this example, we assume no dispersion in the signal. The signal is a linear function of time. Prior to 10 days, the Delay Time is equal to 1 day. After 10 days, the Delay Time is equal to 5 days. The simulated result is shown below (Info_Delay is the output of the Delay element):
To understand this result, let’s assume that the signal represents some measurement, and before the measurement gets reported, it must pass through a chain of ten individuals. Prior to 10 days, it takes 1 day for the information to move through these individuals. After 10 days, it takes 5 times longer (e.g., perhaps they change from working 5 days per week to one day per week). Any measurement received before 9 days is delayed exactly one day. Any measurement received after 10 days is delayed exactly 5 days. Any measurement received between 9 and 10 days (during the tenth day), is delayed from 1 to 5 days. Measurements received early in the 10th day nearly made it all the way through the chain of individuals before their work rate decreased (the delay time increased). These signals are delayed for a little more than 1 day. Measurements received late in the 10th day were not very far along the chain of individuals before their work rate decreased. These signals are delayed nearly 5 days.

In the absence of dispersion and assuming a constant Delay Time, the output of an Information Delay is simply computed as follows:

\[ \text{Output}(t) = \text{Input}(t - \text{Delay Time}) \]

If the signal is dispersed (or the Delay Time is variable), the solution involves a convolution integral of the form:

\[ \text{Output}(t) = \int_{0}^{t} \text{Input}(\tau) f(t - \tau) d\tau \]

\( f(t) \) is the gamma probability distribution, which is equivalent to (and a generalization of) the Erlang distribution that is frequently used in simulation models. \( f(t) \) represents the probability density of the time of “release” from the delay of an input at time 0:

\[ f(t) = \frac{t^{n-1}e^{-t/\beta}}{\beta^n \Gamma(n)} \]

where:

- \( n \) is the Erlang value (specified by the user);
- \( \beta = D/n; \)
D is the mean delay time; and
\[ \Gamma \] is the gamma function (not the Gamma distribution).

The Erlang distribution is equivalent to the distribution of passage times through a cascaded series of \( n \) mixing cells, each of which has a mean residence time of \( \frac{D}{n} \). The gamma distribution represents the time until the occurrence of \( n \) sequential Poisson-process events, where each event’s random time is represented by an exponential distribution with mean \( \frac{D}{n} \).

The gamma distribution is a generalized version of the Erlang distribution, and does not require \( n \) to be an integer. Note that for \( n=1 \), the distribution is exponential, and for increasing values of \( n \) it becomes less skewed, approaching normality for large \( n \).

The standard deviation of the gamma probability distribution is equal to \( \frac{D}{\sqrt{n}} \).

The degree of dispersion for an Information Delay can be specified in terms of either \( n \) or the standard deviation.

Read more: Modeling Information Delays with Dispersion (page 324).

GoldSim actually solves the convolution integral by first carrying out a transformation of the time axis. This allows for accurate representation of variable Delay Times.

Many business decisions use forecasts of future behavior (e.g., the sales rate for a product) based on past observations. In order to simulate such systems using GoldSim, it is necessary to simulate the forecasting process itself. For example, when modeling a supply chain, it is necessary to simulate how orders are placed. In the real world, these orders are placed not based on the current demand (which may not even be measurable), but on a forecast of future demand.

An Information Delay can be used to represent a very commonly used forecasting method known as exponential smoothing. In exponential smoothing, the forecast is based on an exponentially-weighted average of past observations. Mathematically, this is equivalent to the output of an Information Delay with \( n = 1 \). An example of such a simulated forecast is presented below:

As shown below, the browser view of an Information Delay element shows a single output, and has three inputs (the Signal, the Initial Value, and the length of the Delay:}
Using Delay Elements

Material Delay Elements

Material Delay elements are intended to be used to simulate delays in the physical movement (flow) of material. These delays often have a critical impact in the dynamic behavior of systems.

You would use a Material Delay element to simulate processes like the movement of parts on a conveyor belt, the flow of water through an aquifer, the movement of cars from one location to another, and the movement of letters through the mail system.

Material is conserved as it moves through a Material Delay. In some cases, the material may be dispersed while in transit. For example, if you send 100 letters all at once, they will not be delivered at the same time. Rather, there will be some variability in the time at which they are delivered (i.e., the delay time).

In other cases, the material is not dispersed. If a conveyor belt moves at a fixed speed, there will be no variability in the transit times for items that are placed on the conveyor.

The properties dialog for a Material Delay element looks like this:
A Material Delay requires one or more **Inflows** and a **Delay Time**. You can optionally define the degree of **Dispersion**.

The element has two outputs: the primary output of the Material Delay is the lagged flow (the Outflow), which has the same dimensions and order as the Inflow. The secondary output is the amount of material in transit within the Material Delay.

The best way to understand how to use Material Delays is to examine the behavior of the element in a number of simple examples. A file containing these examples (MaterialDelay.gsm) can be found in the General Examples folder in your GoldSim directory (accessed by selecting **File | Open Example...** from the main menu).

Within the Material Delay properties dialog, two sets of Display Units must be specified for a Material Delay (the **Outflow Units** and the **Quantity Units**), one for each of the element’s outputs. The **Outflow Units** determine the dimensions of the Material Delay’s primary output.

**Note:** GoldSim encourages (but does not require) you to specify Outflow Units for a Material Delay whose dimensions are consistent with a flow of materials. In particular, it expects the units to have dimensions of mass/time, volume/time, energy/time, currency/time, or 1/time. When you first specify the Outflow Units, GoldSim will warn you if your units do not have one of these dimensions. GoldSim will not, however, prevent you from using other units.

**Read more:** [Differentiating Between Material and Information Flow](#) (page 150).

The primary output (the Outflow) can only be specified as a value, but can be specified as a scalar, a vector or a matrix. You can specify whether the Material Delay is a scalar, vector or matrix by pressing the **Type...** button. By default, the output of a new Material Delay element is a scalar, dimensionless value. You can also use Material Delay elements to operate on and create vectors and matrices.

**Read more:** [Using Vectors and Matrices](#) (page 782).

The **Quantity Units** determine the Display Units for the secondary output of the element, the Amount in Transit. The dimensions of this output must be the same as those of the Outflow multiplied by Time. For example, if the Outflow Units have dimensions of Mass/Time, the Quantity Units must have dimensions of Mass.

A Material Delay can have multiple Inflows. You add an Inflow to the element by pressing the **Add Material Inflow** button (the green +). A browser tree showing all of the elements in the model will be displayed.

Note that this tree is organized by containment in the same manner as the main browser. To insert a link from this dialog, you select a specific output object (or an element having a primary output), and then press **OK**. If you press **Cancel**, GoldSim will insert a new blank item. In either case, the dialog will close and the item will be added to the list of Inflows in the properties dialog for the element:
The Inflows must have the same attributes (order and dimensions) as the primary output (the Outflow). Although you can edit any of the input fields after the Inflow is added, the Inflows must all be single links to another element (i.e., you cannot enter an expression here). In addition, the Inflows must be non-negative.

Inflows can be deleted using the Remove Material Inflow button (the red X).

The Initial Outflow to the Material Delay must have the same attributes (order and dimension) as the Outflow.

**Note:** The Initial Outflow must be a number or a link from a static variable (e.g., a constant Data element or a Stochastic).

The Delay Time must have dimensions of time and must be positive.

**Note:** The Delay Time for a Material Delay must be greater than or equal to the timestep. That is, GoldSim cannot delay a signal for a smaller time period than a timestep. If you enter a Delay Time which is less than a timestep, GoldSim internally treats the Delay Time as being equal to a timestep. Delay Times that are less than zero will result in a fatal error.

The Dispersion drop-list provides three choices: “None” (the default), “Erlang n”, and “Std. Deviation”.

If one of the latter two is selected, you must enter a value that quantifies the degree of dispersion to the right of this field. If “Erlang n” is selected, you must enter a dimensionless value greater than or equal to 1. If “Std. Deviation” is selected, you must enter a value with dimensions of time. The value must be greater than or equal to zero and less than or equal to the Delay Time.

If the Material Delay is specified as being an array (i.e., a vector or a matrix), you can specify whether the Delay Time and Dispersion are defined as scalars (with the same value being applied to all items of the array), or as arrays (with different values applied to each item of the array). This is done via the Use scalar delay time and dispersion checkbox. If this box is checked (the default), the Delay Time and Dispersion must be entered as scalars. If the box is cleared (which is only possible if the element is specified as an array), the Delay Time and Dispersion must be entered as arrays.

The simplest use of a Material Delay element is to represent the delay of a non-dispersed material flow. To simulate such a system, the Dispersion must be set to “None”.

In this case, the Outflow at any time t is simply the Inflow lagged by the Delay Time:

\[ \text{Outflow}(t) = \text{Inflow}(t - \text{Delay Time}) \]

An example of such a delay is an assembly line that moves at a constant speed (regardless of the rate at which material enters the line).
In the example shown below, the rate changes as a step (at 10 days), and it takes exactly 5 days to traverse the line:

![Material Delay with No Dispersion](image)

In the somewhat more interesting example shown below, the rate of addition to the line varies continuously (as a sine curve with a period of 15 days). Again, it takes 5 days to traverse the line:

![Material Delay with No Dispersion](image)

**Modeling Material Delays with Dispersion**

In many cases, a material flow is dispersed, “smoothed” or “smeared” such that the output represents a weighted average of previous values for the input signal. (With no dispersion, the weight for the input value at \( t - \text{Delay Time} \) is one, and all other previous inputs have a weight of zero).

To specify that the material is dispersed, you must select either “Erlang n” or “Std. Deviation” from the *Dispersion* drop-list. These are two alternative ways to quantify the degree of dispersion while the flow traverses the delay.

**Read more:** [Mathematics of Material Delays](page 339).

If the “Erlang n” is selected, you must enter a dimensionless value greater than or equal to 4. As \( n \) increases, the degree of dispersion decreases. As \( n \) goes to
Using Delay Elements

infinity, the dispersion goes to zero. The maximum amount of dispersion allowed is represented by \( n = 4 \).

If “Std. Deviation” is selected, you must enter a value with dimensions of time. The value must be greater than or equal to zero and less than or equal to the half of the Delay Time. As the Std. Deviation decreases, the degree of dispersion decreases. When the Std. Deviation goes to zero, the dispersion goes to zero. The maximum amount of dispersion allowed is represented by \( \text{Std. Deviation} = 0.5 \times \text{Delay Time} \).

The Erlang n and the Std. Deviation are related by the following equation:

\[
 n = \left( \frac{\text{Delay Time}}{\text{Std. Deviation}} \right)^2
\]

The figure below shows the response of a Material Delay to a step function at ten days for various values of \( n \):

Specifying Initial Outflows for Material Delays

A Material Delay can be assigned an Initial Outflow (the default is zero). The ability to do so is important since the actual processes that you will be simulating may not necessarily start when your simulation begins.

In many cases, the process may have been ongoing for some time prior to the start of the simulation. As a result, your Material Delay (e.g., an assembly line, an aquifer, a pipe) probably already contains some material (and is outflowing material) at the start of the simulation.

GoldSim allows you to specify the **Initial Outflow** for a Material Delay. The Initial Outflow is related to the initial amount of material in transit by the following equation:

\[
 \text{Initial Outflow} = \frac{\text{Initial Amount in Transit}}{\text{Initial Delay Time}}
\]

Material Delays with Time-Variable Delay Times

In some cases, the Delay Time for your Material Delay may change as a function of time. For example, suppose a conveyor moves dirt from one location to another. It is loaded at a constant rate of 2E6 kg/day. The conveyor moves rather slowly, so that it takes 2 days for dirt which is loaded at one end to be offloaded at the other end. Let’s further assume that at 10 days, the conveyor...
Using Delay Elements

breaks down, and it takes 3 days to fix it. The figure below plots the Outflow Rate and the Amount in Transit for this simple model:

To understand this result, let’s first examine the Outflow curve. Prior to time = 10 days, the Outflow Rate is constant and equal to the Inflow Rate (i.e., it is at steady state). At time = 10 days, the conveyor shuts down, and hence the Outflow goes to zero. After time = 13 days, the conveyor starts up again, and the Outflow resumes its original rate. At time = 15 days, however, a large spike of material outflows, and then the Outflow returns to the steady state value again.

Where did this spike come from?

To understand this, examine the Amount in Transit output. Notice that while the conveyor is stopped, the Amount in Transit continues to increase. This is because the Inflow Rate has remained constant at 2E6 kg/day. This mass has to go somewhere (Material Delays conserve mass), so it is “piled up” at the start of the conveyor. Conceptually, it is as if the conveyor was continuing to be loaded, even though it was no longer moving (i.e., a large pile was growing on the first section of the conveyor). It is this “pile” that is released as a spike at time = 15 days (the height of the spike is actually off the scale shown here). Once the conveyor restarted, it took 2 days for the pile to traverse the conveyor.

Of course, this simple example is not very realistic. In reality, they would stop loading the conveyor (e.g., they would pile it next to the conveyor). This is true, and in that case, we would need to control the Inflow Rate. In the simple model above, the Inflow Rate was assumed to remain constant in order to explain the functionality of the element. A more realistic way to represent this particular system is to represent an inflow limit for the Material Delay (that might change with time). In this example, the inflow limit would go to zero when the conveyor stopped. The manner in which you could implement this is discussed below.

**Representing a Material Delay with an Inflow Limit**

In some situations, a Material Delay may have a physical limit with regard to the Inflow rate that it can accept. For example, an assembly line has a maximum rate at which it can accept and process parts. A road also has a maximum rate at which cars can move along it. A conveyor belt has a maximum rate at which it
can accept material (e.g., it will have some weight limit), and when it stops moving, this maximum rate goes to zero.

These kinds of systems can be easily modeled by combining a Reservoir (or a Pool) with a Material Delay (the example discussed here assumes a Reservoir). The Inflow Rate is specified as the Addition Rate for the Reservoir. The maximum rate at which the Material Delay can accept material is specified as the Withdrawal Rate from the Reservoir. The Withdrawal Rate output of the Reservoir then becomes the Input Rate for the Material Delay. In this case, the Reservoir element represents a “pile” at the start of the Material Delay. If the Inflow Rate is greater than the rate at which the Delay can accept material, the pile grows; otherwise it shrinks (and may disappear completely).

As an example, consider a conveyor that moves dirt from one location to another. It is loaded at a constant rate of 2E6 kg/day. The conveyor moves rather slowly, so that it takes 2 days for dirt which is loaded at one end to be offloaded at the other end. Let’s further assume that at 10 days, the conveyor breaks down, and it takes 3 days to fix it. An example of how such a structure can be used to more realistically model this simple system is included in the file MaterialDelay.gsm which can be found in the General Examples folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

The result looks like this:

In this case, while the conveyor is stopped, material accumulates in the Pile (it is not placed on the conveyor). The size of the Pile increases while the conveyor is stopped, and then remains constant (since inflow to the Pile = outflow from the Pile) once the conveyor restarts.

In some cases, it may be necessary to represent a recirculating system (e.g., a system in which water or some other material recirculates in a loop). Under some circumstances, if you try to create such a system, GoldSim may prevent you from “closing” the loop by reporting that the system is recursive. GoldSim can represent such looping systems, but in order to do so, the loop must contain a state variable. A Material Delay can be used to model such a recirculating system by providing a mechanism to “close” the feedback loop.
Read more: [Evaluating Feedback Loops](#) (page 346).

This is best illustrated by examining a simple example. Consider a system consisting of two ponds. The first pond (Pond1) has an initial volume (Initial Value) of 1000 m³ and a Withdrawal Request (due to pumping) of 10 m³/day:

![Reservoir Properties: Pond1](image)

The second pond (Pond2) accepts the Withdrawal_Rate output from Pond1 as an Addition and has an Upper Bound of 400 m³:

![Reservoir Properties: Pond2](image)
Read more: Using the Withdrawal Rate Output of a Reservoir (page 238).
Because Pond2 has an Upper Bound, it has an Overflow_Rate output.

Read more: How a Reservoir Computes the Overflow Rate (page 242).

What we would like to do is route the overflow from Pond2 back into Pond1, hence creating a recirculating system: water is pumped from Pond1 into Pond2, and when Pond2 overflows, it is directed back into Pond1. Logically, there is nothing at all wrong with this system. However, if you try to do this (by routing the Overflow_Rate from Pond2 as an Addition for Pond1, the following error is displayed:

GoldSim is unable to create this link, because it would form a loop that contains no state variables. All loops in GoldSim must contain a state variable. (There are no state variables in the loop because neither the Withdrawal_Rate output nor the Overflow_Rate output is a state variable.)

Fortunately, we can address this by inserting a Material Delay element into this loop. In particular, we can define the Overflow_Rate from Pond2 as the input to a Material Delay:
We can then define the output for the Material Delay as the Addition to Pond1, thereby closing the loop:
This works because the output of a Material Delay is a state variable.

*Read more:* Understanding State Variables in GoldSim (page 341).

But what does this mean physically? It simply means that when water overflows from Pond2 it does not enter Pond1 immediately. There is a delay while it is transported there. In fact, we could also add a Material Delay between Pond1 and Pond2, as there is likely some kind of delay there also. What is the appropriate delay time? If we leave it at zero, GoldSim will use the timestep length as the delay time. That is, material cannot move through a Material Delay in less than a timestep. This would be appropriate if the actual travel time was quite short. (This introduces a slight error if the actual delay time is significantly less than a timestep. If this is important to represent, it would be necessary to shorten the timestep). If the travel time is greater than a timestep, the actual travel time should be entered.

**Mathematics of Material Delays**

In the absence of dispersion and assuming a constant Delay Time, the output of a Material Delay is simply computed as follows:

\[
\text{Outflow}(t) = \text{Inflow}(t - \text{Delay Time})
\]

If the flow is dispersed (or the Delay time is variable), the solution involves a convolution integral of the form:

\[
\text{Outflow}(t) = \int_{0}^{t} \text{Inflow}(\tau) \cdot f(t - \tau) \, d\tau
\]

\(f(t)\) is the gamma probability distribution, which is equivalent to (and a generalization of) the Erlang distribution that is frequently used in simulation models. \(f(t)\) represents the probability density of the time of “release” from the delay of an input at time 0:

\[
f(t) = \frac{t^{n-1}e^{-t/\beta}}{\beta^n \Gamma(n)}
\]

where:

- \(n\) is the Erlang value (specified by the user);
- \(\beta = D/n\);
- \(D\) is the mean delay time; and
- \(\Gamma\) is the gamma function (*not* the Gamma distribution).

The Erlang distribution is equivalent to the distribution of passage times through a cascaded series of \(n\) mixing cells, each of which has a mean residence time of \(D/n\). The gamma distribution represents the time until the occurrence of \(n\) sequential Poisson-process events, where each event’s random time is represented by an exponential distribution with mean \(D/n\).
The gamma distribution is a generalized version of the Erlang distribution, and does not require \( n \) to be an integer. Note that for \( n=1 \), the distribution is exponential, and for increasing values of \( n \) it becomes less skewed, approaching normality for large \( n \).

The standard deviation of the gamma probability distribution is equal to \( \sqrt{n} \). The degree of dispersion for a Material Delay can be specified in terms of either \( n \) or the standard deviation.

Read more: Modeling Material Delays with Dispersion (page 332).

GoldSim actually solves the convolution integral by first carrying out a transformation of the time axis. This allows for accurate representation of variable Delay Times.

The Amount in Transit within a Material Delay is computed by taking the integral of the difference of the Inflow and the Outflow:

\[
\text{Amount in Transit}(t) = \int_0^t \text{Inflow}(\tau) - \text{Outflow}(\tau) \, d\tau
\]

As shown below, the browser view of a Material Delay element shows two outputs, and has at least three inputs (an Inflow, an Initial Outflow, and the length of the Delay):

Note: Element inputs and outputs are only shown in the browser if you choose to Show Element Subitems (accessed via the browser context menu by right-clicking in the browser).

Additional inputs appear for each Inflow that is added to the element.

How GoldSim Carries Out its Calculations

When getting started with GoldSim and/or when building simple models it is not necessary to understand the details of how GoldSim actually carries out its calculations. Hence, if you are new to GoldSim, you probably don’t need to be concerned with these details (and can skip the sections below).

However, to fully take advantage of all of the powerful features in GoldSim and use them in an appropriate manner, all users should eventually obtain a basic understanding of how GoldSim does these calculations. This knowledge not only provides you with a better understanding of the basic assumptions (and hence limitations) of GoldSim, but also allows you to design models that better represent the system you are trying to simulate.
Within a GoldSim model (actually, any dynamic model), there are fundamentally two types of outputs:

1. Outputs whose current value only depends on the current value of their element’s inputs (i.e., they have no “memory”). These outputs themselves fall into one of two sub-categories:
   - Those that represent unchanging input data (or direct functions of input data) and are therefore static, in that they only need to be computed at the beginning of the simulation and do not need to be recalculated every timestep. Examples include the outputs of Data elements and Stochastics (which are not being resampled).
   - Those whose inputs change with time, and therefore need to be recalculated every timestep. Examples include the outputs of Expressions and Selectors whose inputs vary with time.

2. Outputs whose value is computed based on the historical value of the element’s inputs (as opposed to only being a function of the current value of the element’s inputs). These outputs can be thought of as having “memory” of what has happened before. The primary output of a Reservoir or a Pool is a classic example.

This second type of output is referred to as a state variable in GoldSim. State variables are typically the key system variables that you are interested in predicting. Together, the state variables define the predicted state of the system at any time in the future. Examples of state variables include the volume of water in a pond, the amount of money in an account, the status (on/off/failed) of a piece of equipment, and the number of times a certain event has occurred.

All state variables have, by definition, an initial value (e.g., for Integrators, Reservoirs and Pools, the initial value is explicitly specified). This allows the outputs to be computed when there are no historical input values available (e.g., at the start of simulation).

All state variables have “memory”; that is, their value at time $t$ is a function of their value at time $t - \Delta t$. Furthermore, state variables can change during a simulation in two different ways:

- Some state variables are intrinsically a function of time. That is, every timestep, they may change because they are inherently defined with respect to time. Integrators, Reservoirs and Pools are examples of this. Since these elements fundamentally solve time integrals, whenever the simulation time changes, by definition they must be updated.

- Some state variables can be triggered by or respond to other elements in the model, and in turn, change their value. A Status element is a good example of this.

Some state variables may change due to just one of these mechanisms, while others may potentially change due to both. For example, Status elements can only change by the second mechanism; Reservoirs can change due to both.

**Read more:** [The Causality Sequence and Element Updating](page 343).

Note that based on whether or not a particular state variable can change due to the second mechanism listed above, it falls into one of these two categories:

1. Its value at time $t$ is completely determined by its value (and the inputs) at time $t - \Delta t$.
2. Its value at time $t$ is determined by its value (and the inputs) at time $t - \Delta t$, but can also be further modified by its inputs at time $t$. 

---

**Understanding State Variables in GoldSim**

[361x758]How GoldSim Carries Out its Calculations
Most state variables fall into the second category. The most common state variables are the primary outputs of stocks (Integrators, Reservoirs and Pools). A number of other elements, however, also have outputs which are state variables, and these are summarized in the table below.

<table>
<thead>
<tr>
<th>Element</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integrator</td>
<td>Current Value</td>
</tr>
<tr>
<td>Pool</td>
<td>Current Value (Quantity)</td>
</tr>
<tr>
<td></td>
<td>Is_Full</td>
</tr>
<tr>
<td>Reservoir</td>
<td>Current Value</td>
</tr>
<tr>
<td></td>
<td>Is_Full</td>
</tr>
<tr>
<td>Stochastic</td>
<td>Current Value</td>
</tr>
<tr>
<td></td>
<td>Probability Density</td>
</tr>
<tr>
<td></td>
<td>Cumulative Probability</td>
</tr>
<tr>
<td>Script</td>
<td>Result (primary output)</td>
</tr>
<tr>
<td>Extrema</td>
<td>Current Value</td>
</tr>
<tr>
<td>Status</td>
<td>Current Status</td>
</tr>
<tr>
<td>Decision</td>
<td>Last Decision</td>
</tr>
<tr>
<td>Milestone</td>
<td>Completion Status</td>
</tr>
<tr>
<td></td>
<td>Date of Completion</td>
</tr>
<tr>
<td></td>
<td>Elapsed Time of Completion</td>
</tr>
<tr>
<td>Timed Event Generator</td>
<td>Cumulative Number Emitted</td>
</tr>
<tr>
<td>Triggered Event Generator</td>
<td>Cumulative Number Emitted</td>
</tr>
<tr>
<td>Discrete Change</td>
<td>Cumulative Number Emitted</td>
</tr>
<tr>
<td>Information Delay</td>
<td>Current Value</td>
</tr>
<tr>
<td>Material Delay</td>
<td>Current Value</td>
</tr>
<tr>
<td></td>
<td>Amount in Transit</td>
</tr>
<tr>
<td>Event Delay</td>
<td>Cumulative Number Emitted</td>
</tr>
<tr>
<td></td>
<td>Number in Transit</td>
</tr>
<tr>
<td></td>
<td>Number in Queue</td>
</tr>
<tr>
<td></td>
<td>Mean Time for Emitted Events</td>
</tr>
<tr>
<td></td>
<td>Current Service Time</td>
</tr>
<tr>
<td>Discrete Change Delay</td>
<td>Cumulative Number Emitted</td>
</tr>
<tr>
<td></td>
<td>Number in Transit</td>
</tr>
<tr>
<td></td>
<td>Number in Queue</td>
</tr>
<tr>
<td>Conditional Container</td>
<td>Activity Status</td>
</tr>
<tr>
<td></td>
<td>Completion Status</td>
</tr>
<tr>
<td>Previous Value</td>
<td>Previous value of input</td>
</tr>
<tr>
<td>Time Series</td>
<td>Current Value</td>
</tr>
<tr>
<td>History Generator</td>
<td>Current Value</td>
</tr>
</tbody>
</table>

Outputs listed in **bold** fall into category #1 above (i.e., they can never be impacted by inputs from the current timestep).
The icons for state variables are indicated in browsers (e.g., output ports) by showing a vertical line to the left of the output triangle:

Reservoir1 (the primary output) and Is_Full are state variables.

**Note:** In addition to the elements listed above, a number of elements in GoldSim’s extension modules also have outputs that are state variables.

### The Causality Sequence and Element Updating

In order for GoldSim to determine how to carry out its calculations, it first must create the *causality sequence* for all of the elements. This represents the sequence in which the elements must be logically computed. For example, if A was a function of B, and B was a function of C, then C would be sequenced first, followed by B, and then followed by A. Although the sequence is obvious in this trivial example, in complex models the correct causality sequence may not be as obvious.

The internal logic within GoldSim for updating a model based on its causality sequence is very complex, and you don’t generally need to be concerned with it. However, to better understand the basic concepts of the causality sequence and how GoldSim actually updates a model every timestep, let’s consider another example that is a bit more complex:

In this model, we are computing the volume of water in a pond (represented by a Reservoir). The outflow (Withdrawal Request) from the pond (Pumping_Rate) is an Expression that is defined as a function of time. Inflow1, Inflow2, and therefore Total_Inflow are constant values. The Water_Level is a function of the Volume (the function being defined by a Lookup Table). So what does the causality sequence look like for this model?
We can actually view the causality sequence by pressing F10, which brings up the following dialog:

![Causality sequence for Model Root dialog](image)

**Read more:** Viewing and Modifying the Causality Sequence (page 968).

What should immediately catch your attention here is that four of the elements (Stage_Storage, Inflow1, Inflow2 and Total_Inflow) do not appear in the list. This can be explained by pointing out that there are actually two different sequences: the Static Sequence and the Function Sequence. The Static Sequence consists of those elements that only need to be computed once (at the beginning of the simulation), since they cannot change with time. The Function Sequence consists of those elements that can change as a function of time.

By default, the dialog displays the Function Sequence. If we choose to view the Static Sequence (in the top right corner of the dialog), it looks like this:

![Causality sequence for Model Root dialog (Static Sequence)](image)

These three elements cannot change with time, and only need to be computed once. Note, however, that the sequence indicates that Inflow1 and Inflow2 must logically be computed before Total_Inflow.

**Note:** The Lookup Table element Stage_Storage is a special case in that it does not appear in either list. This is because it actually does not compute an output itself and hence never really needs to be updated; it is used to define a static function (completely analogous to a built-in function such as sin or max) that is used by other elements (in this case, Water_Level). Hence, Lookup Tables do not appear in the causality sequence.

Returning to the Function Sequence, let’s walk through how GoldSim actually carries out its calculations every timestep for this example. As noted above, at the beginning of the simulation, GoldSim first computes all of the static elements (i.e., those elements in the Static Sequence). After doing so, it does not have to compute these again for the rest of the simulation.

Every timestep, GoldSim carries then out its calculations in three steps:

1. It changes the system clock to the appropriate time (e.g., moves it forward a timestep).
2. It computes those elements that have state variable outputs that are intrinsic functions of time (e.g., Reservoirs). In the discussion below, this is referred to as the *intrinsic update* of the element.
3. It computes the elements in the Function Sequence.

So prior to evaluating the Function Sequence, every timestep, after changing the system clock, GoldSim first updates the state variable outputs for elements that are intrinsic functions of time. In this example, the only output that falls into this category is the primary output (the current value) of the Reservoir element named Volume. Note that to do this, GoldSim uses the Additions and Withdrawal Requests that were computed at the previous timestep. (This is because Additions and Withdrawal Requests are forward-looking; they represent the values over the next timestep).

It then computes the Function Sequence. First it updates the Pumping_Rate.

It then updates the Reservoir again. In this particular model, however, other than the intrinsic changes (represented by the Additions and Withdrawal Requests) there are no other elements that impact the primary output of the Reservoir, so it is not changed by the Function Sequence calculation. However, other outputs of the Reservoir that are not state variables (e.g., the Withdrawal_Rate) are updated at this time. Moreover, the inputs for the Reservoir (e.g., its Withdrawal Request input) are updated (e.g., due to a change in the Pumping_Rate) and “remembered” so that they can be used for the next intrinsic update the following timestep.

Finally, it computes the Water_Level (based on the current value of the Volume).

Note: Outputs that are intrinsic functions of time may actually be evaluated twice under some circumstances within the same timestep: first in step #2 (e.g., for the case of a Reservoir, to bring it up to the current time via its intrinsic behavior of time integration of additions and withdrawals), and then again in step #3 (e.g., in the case of a Reservoir, if an Upper Bound changes, forcing the output to a certain value).

Several additional points should be noted regarding the causality sequence:

- The three steps above are not only carried out during scheduled timesteps. They are also carried out during unscheduled timesteps that GoldSim automatically inserts to more accurately represent the system, and in some cases, may be carried out multiple times within the same timestep (e.g., within a looping Container).

  Read more: Understanding Timestepping in GoldSim (page 455).

- Looping systems obviously complicate the causality sequence. In some cases, GoldSim can handle these systems automatically. In other cases, you may need to take action in order to represent logic that is recursive.

  Read more: Evaluating Feedback Loops (page 346).

- For some complex looping systems (particularly those involving events that impact state variables), there may exist more than one way to sequence that system (and these may produce different results). In these cases, you can manually force a particular causality sequence to ensure that the model accurately represents your system.

  Read more: Invalid and Ambiguous Causality Sequences (page 349).

- When Discrete Changes are present in a Function Sequence, the Discrete Changes are propagated to any affected stock elements.
How GoldSim Carries Out its Calculations

instaneously when the Discrete Change is updated in the sequence (regardless of where the affected stocks are in the sequence).

Read more: Discrete Change Elements (page 383).

Evaluating Feedback Loops

Simple models have a direct chain of causality: input data affect some elements, which affect other elements, and so on, until eventually the elements that calculate the desired results of the model are reached:

GoldSim automatically analyzes your entire model to identify "who affects who", and ensures that the "upstream" elements are calculated prior to the "downstream" elements. At each timestep the elements are updated in this causality-based sequence.

Read more: The Causality Sequence and Element Updating (page 343).

Most real-world models, however, contain elements whose output can, directly or indirectly, affect one of their own inputs. This creates a looping or circular system such as the one shown below:

These types of systems are referred to as feedback loops. Feedback loops are present in one form or another in most real-world systems, and can be readily modeled in GoldSim. In the example above, A affects B which affects C, and then C “feeds back” to affect A and so on. That is, feedback loops represent a closed chain of cause and effect. Note that the terms “feedback” and “cause and effect” intentionally imply that the relationship between the variables is dynamic and the system changes over time (although systems with feedback loops can also reach a dynamic equilibrium).

GoldSim allows you to create looping systems like this. In order to do so, however, the loop must meet one requirement: it must contain at least one state variable. State variables provide inertia or “memory” to a system, and always have an existing value prior to updating the model at a new time point. As a result, they allow a looping model to properly initialize itself and subsequently move from one timestep to the next.

Read more: Understanding State Variables in GoldSim (page 341).

In the simple looping example above, A references the primary output (a state variable) of C. Let’s also assume that B represents the Rate of Change (an addition) to C.

Every timestep, after changing the system clock, GoldSim first updates the state variable outputs that are intrinsic functions of time. In this example, the only element that falls into this category is the Reservoir named C. Its primary output is a state variable. So the primary output for C is first brought up to the current time (based on the value of B at the end of the previous timestep).

After doing this, GoldSim then updates the elements according to the causality sequence. If we were to view the causality sequence for the system above, it would look like this:
So first GoldSim computes A (based on the updated value of the primary output of C). Then it computes B. Finally, GoldSim updates C again.

Why is C updated twice (prior to evaluating the causality sequence and then again at the end of the causality sequence)? In this simple example, it is because C has multiple outputs, not all of which are state variables (e.g., the Withdrawal_Rate). These non-state variables need to be updated here at the end of the causality sequence. Moreover, during this update, C stores the new input for B for use the next time the clock is advanced.

Note: In some cases, GoldSim may not be able to create a valid sequence at all. This is the result of the system having circular (recursive) logic without a state variable involved. GoldSim can still represent such systems, but in order to do so, it is necessary for you to explicitly add a state variable to the system.

Read more: Invalid and Ambiguous Causality Sequences (page 349).

Feedback loops are present in one form or another in most real-world systems. There are two basic kinds of feedback loops: positive feedback loops and negative feedback loops.

- **Positive feedback loops** are self-reinforcing. They generate growth and amplify changes. The more adult rabbits you have, the more baby rabbits that are produced; the more baby rabbits that are produced, the more adult rabbits you have, and so on until the world is full of rabbits (or this positive loop is counteracted by a negative feedback loop).

- **Negative feedback loops** are self-correcting. They drive systems toward equilibrium and balance. The more rabbits you have, the less food you have; the less food you have, the less rabbits you have.

The dynamics of most systems are driven by the interactions of many such loops.

The example below shows two simple feedback loops: the positive feedback loop associated with earning interest on a bank account; and the negative feedback loop associated with leakage from a pond. Note that both loops contain a dynamic element (in this case, a Reservoir):
This model (Feedback.gsm) can be found in the General Examples folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

SubSystems are treated by other elements as if they were a single element (analogous to a complex Expression element). This has an important implication for systems that include feedback loops. In particular, if a SubSystem contains elements with state variable outputs, and these state variable outputs are referenced outside of the SubSystem, they will not be treated as state variables outside of the SubSystem.

Read more: Treating a Container as a SubSystem (page 135).

As a result, if you try to create a feedback loop by referencing a state variable contained within the SubSystem outside of the SubSystem, GoldSim will not be able to create the loop (and will provide an error message indicating that creating such a link would result in a recursive system).

In most cases, the most appropriate way to “close” such a loop is to reference the state variable from the SubSystem in a Material Delay element (whose primary output is a state variable) located outside of the SubSystem.


When using the Function Of or Affects View for an element that is within a feedback loop, GoldSim will stop building a branch of the dependency tree as soon as an item is repeated (and it will mark this as a “loop”).

Read more: Viewing Element Dependencies (page 112).

For example, if Reservoir1 is a function of X, X is a function of Y, and Y is a function of Reservoir1, the Function Of View for Reservoir1 would look like this:
When using the Find function (Ctrl+F) in a Function Of or Affects view, one of the options is to search in Labels. If this is selected, and you enter “loop”, GoldSim will find the next loop in the list (since it will find the word “loop” in the label for the element).

Read more: Finding Elements (page 112).

When creating complex models, determining the appropriate causality sequence can sometimes become difficult. In particular, two problems can sometimes occur:

- GoldSim may not be able to create a valid sequence at all. This is the result of the system having circular (recursive) logic without a state variable involved. If you tried to build such a model, GoldSim would be unable to create a valid causality sequence, and would display an error, warning you that the system is recursive. GoldSim can still represent such systems, but in order to do so, it is necessary for you to explicitly add a state variable to the system. Depending on the system, there are generally two ways to address this issue:
  - In some cases, the system does in fact represent a feedback loop, but due to the lack of a state variable, you are unable to “close” the loop. These systems can usually be addressed by adding a state variable in the form of a Material Delay.
    
  - In other cases, the system does not represent a feedback loop, but instead represents a coupled process (that may not be dynamic at all). These systems can be solved by using Previous Value elements to close the recursive loop.
    
    Read more: Creating Recursive Loops Using Previous Value Elements (page 960).

- In rare cases, GoldSim can sequence the system, but there may be more than one valid way to sequence the system, and these produce different results. This can occur, for example, if state variables that are impacted by an event (e.g., a discrete change) affect another element. Should the downstream element be updated before or after the state variable is modified? From the point of view of GoldSim, both sequences for the downstream element may be equally valid, but depending on the application, you may specifically require one instead of the other. In these cases, you can explicitly force a particular causality sequence to ensure that the model accurately represents your system.
    
    Read more: Addressing Ambiguous Causality Sequences (page 971).
Chapter 5: Simulating Discrete Events

We are ready for any unforeseen event that may or may not occur.

Dan Quayle

Chapter Overview

Most of the elements in GoldSim (e.g., Reservoirs, Expressions, Time Series elements) were designed primarily to be used to represent continuous variables and processes. In many systems, however, processes occur that are discrete as opposed to continuous. These discrete occurrences are referred to within GoldSim as events. GoldSim provides powerful capabilities for superimposing the occurrence and effects of discrete events onto continuously varying systems.

This allows for the realistic simulation of things such as financial transactions, accidents, system failures, storms, labor strikes, and lawsuits. Events such as these have important effects on the performance of many systems, and it is therefore important to represent them in a realistic manner.

This chapter describes the GoldSim elements that you can use to simulate the occurrence and consequences of discrete events.

In this Chapter

The following topics are discussed in this chapter:

- Basic Concepts of Discrete Event Modeling
- Understanding Event Triggering
- Generating Discrete Event Signals
- Responding to Events
- Generating Discrete Changes Using Time Series Elements
- How GoldSim Inserts Events into a Simulation
- Determining if an Event Has Occurred
- Controlling the Calculation Sequence of Events
Basic Concepts of Discrete Event Modeling

A discrete event is something that occurs instantaneously (as opposed to continuously or gradually) in time. It represents a “spike”, a discontinuity, or a discrete change of state for the system.

The flow of water out of a hole in a bucket is a continuous process (as long as water remains in the bucket). Puncturing the bucket to create the hole is a discrete event. Stopping the flow from the bucket (by plugging the hole or by the bucket becoming empty) is also a discrete event. Similarly, through continuous compounding of interest, the money in a bank account continuously increases, but the account can also increase and decrease instantaneously due to discrete events (i.e., deposits and withdrawals).

Of course, “instantaneous” and “gradual” are relative terms. That is, whether something is treated as instantaneous or gradual is a function of the time scale of interest, and hence you must differentiate between the two based on the context of your model. Typically, the differentiation will be obvious. For example, if the time scale of interest is 10 years, something happening over the span of a day can be considered to be “instantaneous”. If the time scale of interest is several days, however, something happening over the span of a day would in most cases need to be treated in a continuous manner.

GoldSim handles “instantaneous” changes to a model by providing a mechanism for a model to generate and respond to discrete events. This is accomplished by providing the ability to instantaneously trigger an element to take a particular action (e.g., instantaneously change its value) in response to an event. Hence, in GoldSim, an event is specifically defined as an instantaneous occurrence that subsequently triggers a particular action.

In GoldSim, an event can be generated in one of four ways:

- The event occurs when a specified condition (e.g., \( X > Y \)) becomes true or false;
- The event occurs when a specified output in the model changes;
- The event occurs at a specified calendar or elapsed time; or
- The event occurs based on a specified rate of occurrence, which can be treated as regular or random ("occur exactly once a week" or "occur, on average, once a week").

In addition, some elements can respond to an event generated via one of the mechanisms above, and generate a new event.

**Note:** Depending on how the event was generated, it may not fall exactly on a “scheduled update” (i.e., a timestep that was defined in the **Time** tab of the Simulation Settings dialog). That is, an event could actually occur between scheduled updates of the model. Such events can trigger an “unscheduled update” of the model.

**Read more:** [Understanding Timestepping in GoldSim](#) (page 455).

A variety of GoldSim elements can be triggered by an event, with each element responding to the event (taking a particular action) in a different manner. For example, the action taken by a Stochastic element when it is triggered by an
event is to resample itself. The action taken by a Status element when it is triggered by an event is to instantaneously change its value (from True to False or from False to True).

The manner in which events are specified to occur is discussed in detail in “Understanding Event Triggering” and “Generating Discrete Event Signals”. The elements that can respond to events, and the manner in which they do so, are discussed in “Responding to Events”.

Read more: Understanding Event Triggering (page 355); Generating Discrete Event Signals (page 364); Responding to Events (page 383).

In some cases, an event will occur (e.g., X becoming greater than Y) which triggers a particular action in a single element (e.g., change the output of a Status element to True). In such a case, the event is internal to that element, and it does not directly impact other elements. In other cases, however, an event may impact multiple elements, or one element may respond to an event by triggering other elements to take a particular action. In these cases, it is necessary for discrete signals to propagate between elements.

If you think carefully about this, you will realize that this is very different from how information is typically passed between elements in GoldSim. In most situations, information is transferred between elements (via links) continuously in time. For example, if Expression element X is defined as being equal to A + B, the values of A and B are continuously sent to X (i.e., the information is “broadcast” through the link throughout the simulation). Conceptually, at any given time in the simulation, X is receiving a signal from A and B.

In order to propagate events (and their consequences) between elements in a model, however, it is necessary to send information between elements intermittently as a "spike" or discrete "packet" of information. To facilitate this, GoldSim allows certain elements to emit and receive (i.e., produce as outputs and/or accept as inputs) a discrete signal. Discrete signals are a special category of outputs that emit information discretely, rather than continuously.

When you view an element that can emit or receive a discrete signal in the browser, the symbols for the inputs and outputs that represent discrete signals are identified as red triangles (as opposed to blue, green or gray for other output types).

Note: Element inputs and outputs are only shown in the browser if you choose to Show Element Subitems (accessed via the browser context menu by right-clicking in the browser).

Within GoldSim, there are actually two types of discrete signals that can be passed from one element to another: discrete event signals and discrete change signals.
A **discrete event signal** is a discrete signal indicating that something (e.g., an earthquake, a monthly bank deposit, a traffic accident) has occurred. It does not describe the consequence of that occurrence; it simply emits a signal between elements indicating that an event has occurred.

A **discrete change signal**, on the other hand, emits information regarding the response to an event. In particular, a discrete change signal contains two pieces of information: a value (e.g., 10 dollars) and an instruction (e.g., Add). A discrete change signal can only be generated in response to an event. It can only be received by a few select elements (e.g., Reservoirs, some Financial Module elements) which understand how to process it.

Only certain elements in GoldSim can emit (as outputs) or receive (as inputs) discrete signals. These elements and the manner in which they can interact with discrete signals are summarized in the tables below.

### Elements that Emit and Receive Discrete Event Signals

<table>
<thead>
<tr>
<th>Element</th>
<th>Emits Discrete Event Signals</th>
<th>Receives Discrete Event Signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timed Event</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Triggered Event</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Event Delay</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Decision</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Random Choice</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Previous Value</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Stochastic</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Milestone</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Discrete Change</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Extrema</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Interrupt</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

In addition to being triggered by discrete event signals, the elements indicated in the right-hand column can also be triggered by discrete change signals (which are treated as discrete event signals when entered into a field that accepts discrete event signals).

**Read more:** [Understanding Event Triggering](#) (page 355).

!![Note](#): An additional type of element, a conditional Container, can also emit and receive discrete event signals.

**Read more:** [Using Conditional Containers](#) (page 899).
Elements That Emit and Receive Discrete Change Signals

<table>
<thead>
<tr>
<th>Element</th>
<th>Emits Discrete Change Signals</th>
<th>Receives Discrete Change Signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete Change</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Discrete Change Delay</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Integrator</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Reservoir</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Pool</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Time Series</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Splitter</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Allocator</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Previous Value</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Note: Some elements in the various GoldSim extension modules can also emit and receive discrete signals. The elements listed above only include those found in the basic GoldSim framework.

The manner in which each of these elements can be used to simulate the occurrence and consequences of discrete events is discussed in detail in the sections below.

An example model which demonstrates the basic types of discrete event configurations (DiscreteEvents.gsm) can be found in the General Examples/Events folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu). The folder also includes more detailed examples that utilize discrete event logic.

Understanding Event Triggering

All discrete event modeling involves triggering of one form or another. When you trigger an element, you are telling it that a discrete event has occurred that you want the element to respond to. Therefore, before describing the details of the various elements associated with discrete event modeling, it is first necessary to understand the concept of triggering.

The following element types can be triggered in GoldSim:

- Triggered Event
- Event Delay
- Stochastic
- Status
- Milestone
- Decision
- Random Choice
- Discrete Change
• Extrema
• Conditional Containers

(Specialized elements within some of the GoldSim extension modules, such as the Reliability Module and the Financial Module, can also be triggered.)

Each of these elements responds to a trigger in a different way. They are, however, all triggered in the same way. That is, these elements all share a common Triggering dialog, which controls how they are triggered. The Triggering dialog is accessed via a button in the dialog for each element:

![Trigger button](image)

**Note:** Depending on the element, the label for the trigger button will not always be “Trigger…”.

**Note:** Holding your cursor over the trigger button displays a tool-tip summarizing the current trigger settings.

The Triggering dialog for every element looks like this:

![Triggering dialog](image)

Note that for some elements, an additional button (for specifying Resource interactions) is available on the Trigger dialog:

![Additional button](image)

The Triggering dialog is discussed in detail in the sections below.

**Specifying Triggering Events**

The top part of the Triggering dialog (which is the only portion visible when you first define a trigger) is where you define the triggering events (the discrete occurrences) that you want the element to respond to:
The **More** button provides access to advanced triggering options: specifying precedence conditions and required conditions.

**Read more:** [Specifying a Precedence Condition for a Trigger](page 361); [Specifying a Required Condition for a Trigger](page 362).

By default, when you first create an element that can be triggered, no triggers will exist, and therefore the element will never be triggered.

---

**Note:** There is one exception to this rule. The Deactivation trigger for conditional Containers has a default defined trigger.

**Read more:** [Using Conditional Containers](page 899).

To define a trigger for the element, you simply press the **Add** button to add a row to the list of triggering events:

By default, the **Type** of event added will be “On Event”. If you click on the small button in the **Type** column, a drop-list will be presented allowing you to edit the event type:
There are eight types of events that can be added:

- **On Event**: The Trigger Definition must be a discrete event or discrete change signal from another element. The element is triggered whenever the signal is received.

- **On Changed**: The Trigger Definition can be any continuous output (it cannot be an expression or a discrete signal). The element is triggered whenever the value of Trigger Definition changes.

- **On True**: The Trigger Definition can be any condition output or conditional expression. The element is triggered whenever the Trigger Definition becomes True.

- **On False**: The Trigger Definition can be any condition output or conditional expression. The element is triggered whenever the Trigger Definition becomes False.

- **At Stock Test**: The Trigger Definition must be a conditional expression of the form “A>B”, “A>= B”, “A<B”, “A<=B”, or “A=B” where A is the primary output of a Reservoir, a Pool or a Fund (from the Financial Module). The element is triggered whenever the Trigger Definition becomes True. As discussed below, this triggering event interrupts the clock and adds an unscheduled update.

- **At Date**: The Trigger Definition must be a date or date and time, enclosed in quotations. (Alternatively, the date can also be expressed as the amount of time since 30 December 1899). The element is triggered whenever the simulated date reaches the specified date. As discussed below, this triggering event interrupts the clock and adds an unscheduled update.

- **At ETime**: The Trigger Definition must be an elapsed time. The element is triggered whenever the simulated elapsed time reaches the specified elapsed time. As discussed below, this triggering event interrupts the clock and adds an unscheduled update.

**Read more**: Referencing Dates in Expressions (page 131).

- **Auto Trigger**: An Auto Trigger requires no user-defined Trigger Definition and its behavior is defined by its context (i.e., the type of element). Auto Triggers can react to the activation or deactivation of their parent Container.
In some special cases (e.g., specialized Reliability elements), they can also be used to respond to other types of actions impacting a parent Container (e.g., preventive maintenance). Depending on the type of element, this option may not always be available.

**Note:** At Stock Test, At Date and At ETime triggers interrupt the clock and insert an unscheduled update when they occur (whereas On True, On False and On Changed triggers do not create an unscheduled update). To understand the implications of this, consider an example in which your scheduled updates were every 10 days. There are two different ways you could try to trigger an event when the value of Reservoir A became greater than the value B. You could create an At Stock Test trigger of A > B, or you could create an On True trigger of A > B. If we assume that A > B actually became true at 15 days, these two triggers would behave very differently. The At Stock Test trigger would catch this point exactly, and insert an unscheduled update at 15 days. In the absence of any other unscheduled updates, however, the On True trigger would not be evaluated and implemented until 20 days. Similarly, if you wished to trigger an event at a specific elapsed time (e.g., 17 days), you could try to do so in two different ways. You could trigger the event using an At ETime trigger of 17 days, or you could create an On True trigger with ETime >=17 days. Again, these two triggers would behave very differently. The At ETime trigger would catch this point exactly, and insert an unscheduled update at 17 days. In the absence of any other unscheduled updates, however, the On True trigger would not be evaluated and implemented until 20 days.

**Read more:** Understanding Timestepping in GoldSim (page 455).

**Warning:** Care must be taken if you choose to disable insertion of unscheduled updates while using At Stock Test, At Date or At ETime triggers. These triggers are designed to interrupt the clock and insert a new update, and may not trigger at all if you disable unscheduled updates.

**Read more:** Controlling Unscheduled Updates (page 470).

You can add as many triggering events as desired (with different event types). The element is triggered when any one of the events occurs.

The following details should be noted regarding triggering:

- Unless you are using conditional Containers, you should generally never use an Auto Trigger. If you do, the element will automatically be triggered when its parent Container activates (which in the absence of conditional Containers, is at the beginning of the realization). In most cases, this is not how you will want elements to be triggered.

  **Read more:** Using Conditional Containers (page 899).

- On Event triggers can have multiple triggering signals. These are specified by separating the discrete event or discrete change signals with semicolons. If an On Event trigger has multiple triggering signals, the event is triggered after all of the event signals have occurred. For example, if two triggering signals are specified, one which occurs at 10 days, and a second at 20 days, the event will be triggered at 20 days.
On Changed, On True and On False events are triggered whenever the element determines that the argument to the event has changed, become true, or become false, respectively. If the element is in a conditional Container, it does not evaluate these arguments while it is inactive, and therefore some On Changed, On True and On False events that occur while the Container is inactive can be “delayed” until the element activates (rather than ignored). For example, if an On True trigger is defined as “Etime > 15 days”, and the element is inactive until 20 days, the event will actually be triggered at 20 days (since, from the element’s viewpoint, this is first time that it is able to determine that the condition has become true).

Read more: Using Conditional Containers (page 899).

Unless the Trigger Definition has a well-defined initial value (e.g., such as a Reservoir), On Changed triggers cannot fire at the beginning of a simulation (ETime = 0).

You should never reference an item of an array within an On Changed, On True or On False trigger (e.g., On Changed vector1[A]). If you do, the trigger will fire if any item of the vector changes (or becomes true or false). If you need to do this, create a scalar Expression with the appropriate definition and use that in the trigger.

On True events are triggered when the condition switches from False to True. Conversely, On False events are triggered when the condition switches from True to False. The one exception to this rule is the behavior of these triggers at Etime = 0. In particular, an On True trigger that is true at Etime = 0 (e.g., “Etime >= 0 days”) or an On False trigger that is False at Etime = 0 (e.g., “Etime != 0 days”) will fire at Etime = 0.

You should generally not use equality conditions which reference Time or Etime as a trigger (e.g., an On True triggering event with “Etime = 10 day”). This is because the expression is only evaluated every timestep (and not between timesteps), such that if a timestep does not fall exactly on the time referenced in the condition, the condition will never be True. In such a case, it is always better to use expressions such as “Etime >= 10 day”.

If the argument to the On Changed trigger is a Run Property that conceptually cycles (e.g., Hour), and your timestep is such that the actual value remains unchanged (e.g., Changed(Hour), with a 1 day timestep), GoldSim will still consider that the argument has changed after such a timestep (since the fact that it remains unchanged is simply an artifact of timestepping).

Read more: Understanding and Referencing Run Properties (page 486).

Several elements have a checkbox on the Trigger dialog labeled For simultaneous events, act once. If this box is checked and multiple events occur at the same time, the element will be triggered only once. If the box is cleared, the element will be triggered once for each event. In cases where the triggering of an element multiple times by simultaneous events would not make any difference (e.g., Stochastics), this checkbox is not available in the Triggering dialog, and the simultaneous events only trigger the element once.
The example model Triggers.gsm in the General Examples/Events folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) contains examples of the various types of triggers.

In some cases, it may be convenient to “hold” or delay the triggering of an element until a particular precedence requirement is met. If the triggering event occurs after the precedence requirement is met, the element is immediately triggered. If the triggering event occurs before the precedence requirement is met, the element is triggered as soon as the precedence requirement is met.

To facilitate these kind of triggers, GoldSim allows you to specify a Precedence Condition for a triggering event.

You add a Precedence Condition by pressing the More button on the Trigger dialog to expand it, and then selecting the Then wait for Precedence Condition checkbox. When you do so, the Precedence Condition field becomes available. The field accepts any condition output or conditional expression.

The triggering of the element is “held” until the specified Precedence Condition is met (the Condition becomes True), and then the element is triggered. Hence, unless the Precedence Condition is met at the time that the triggering event occurs, the element will be triggered (if it is triggered at all) after the actual triggering event has occurred.

For example, suppose that you were simulating a project, and you wanted to trigger a particular task to start. Before you can start the task, however, you must first ensure that 1) you received the necessary permission (e.g., a building permit); and 2) you had sufficient money to carry out the task.
To simulate this, you could model the permit process as a triggering event (i.e., obtaining a permit is a triggering event), while the Precedence Condition could be modeled as a simple condition related to the amount of money available. Your Triggering dialog might look like this:

![Triggering dialog](image)

**Note:** If you define multiple triggering events that occur at different times, and you have specified a Precedence Condition, it is possible that both events could occur (at different times), and then upon the Condition becoming true, both would be simultaneously released. In such a case, if **For simultaneous events, act once** is checked, only one event would be released.

### Specifying a Required Condition for a Trigger

In some cases, you may not want to trigger an element unless a particular required condition is met at the time that the triggering event occurs.

To facilitate these kind of triggers, GoldSim allows you to specify a Required Condition for a triggering event.

You add a Required Condition by pressing the **More** button on the Trigger dialog to expand it, and then selecting the **Then check if Required Condition is True** checkbox. When you do so, the Required Condition field becomes available. The field accepts any condition output or conditional expression.
If the condition is True when the triggering event occurs, the element is triggered; otherwise it is not triggered (i.e., the triggering event is ignored).

For example, suppose that you were simulating the performance of a dam, and you defined an event (e.g., a storm) which resulted in the failure of the dam. The dam only failed, however, if at the time the storm occurred, the dam’s spillway was degraded to a certain extent.

To simulate this, you could model the storm as a triggering event, monitor the spillway condition elsewhere in the model, and enter this as the Required Condition that must be met in order for the element to be triggered.

Your Triggering dialog might look like this.

Note: Triggered events are not “held” until the Required Condition is met (as they are for Precedence Conditions). If the condition is not True when the triggering event occurs, the triggering event is ignored (i.e., discarded).

Note: If you specify both a Precedence Condition and a Required Condition, events are evaluated as follows: If the Precedence Conditions is True when the triggering event occurs, the Required Condition is evaluated immediately. If the Precedence Condition is False when the triggering event occurs, the Required Condition is not evaluated until the Precedence Condition become True.

Required Conditions can be used to simulate the equivalent of an If True triggering event (i.e., to trigger something when it is true rather than when it becomes true). To do so, simply use an On Changed trigger with ETime as the argument, and a Required Condition that is the actual If True condition:
A **Resource** is something that has a limited supply (e.g., spare parts, fuel, skilled personnel, money) and is required in order for elements of the modeled system to carry out certain actions.

**Read more:** [Using Resources](page 839).

Only certain kinds of elements in GoldSim can interact with Resources. Some of these elements can interact with Resources through their Trigger dialog. In these cases, a **Resources** button is added to the dialog:

This button allows you to define Resource Requirements for the trigger. These represent special conditions that must be met in order for the trigger to be pulled. If they are not met, the trigger is held (similar to a precedence condition) until the Resource Requirements are met.

---

**Note:** Holding your cursor over the **Resources**... button displays a tool-tip summarizing the current Resource Requirements.

Resources are an advanced feature, and by default, an element has no Resource Requirements.

**Generating Discrete Event Signals**

There are five elements that can generate discrete event signals:

- Timed Events;
• Triggered Events;
• Decisions;
• Random Choices; and
• Event Delays.

These elements are described below.

Timed Event elements produce discrete event signals based on a specified rate of occurrence.

The dialog for a Timed Event looks like this:

![Timed Event Properties dialog](image)

The first item that must be specified for a Timed Event is the **Type**.

The **Type** list box contains five choices:

- **Regular time intervals**. The event occurs exactly according to the specified **Rate**. Hence, if the rate was 0.5 day\(^{-1}\), the event would occur exactly every two days, starting at two days. The **Rate** must have dimensions of inverse time. If desired, you can specify the rate as a function of time such that it changes during the simulation.

- **Random time intervals (Poisson)**. The event is simulated as a Poisson process, such that the *expected* number of occurrences over some time period T is equal to the product of the **Rate** and T. The mathematics of a Poisson process are discussed in more detail in Appendix B. The **Rate** must have dimensions of inverse time. If desired, you can specify the rate as a function of time such that it changes during the simulation.

- **Stochastic element time intervals**. For this event, a **Stochastic** element must be specified (rather than a Rate). The Stochastic element that is specified represents the distribution of the time intervals between events. As such, it must have dimensions of time. The Distribution output of the Stochastic must be referenced:
Note: if the Stochastic is time-dependent, GoldSim uses the distribution from the previous timestep to compute the occurrence of the next event.

**Defined cumulative event count.** This event requires as input a cumulative Count. This represents the cumulative number of events to be emitted. It is generally entered as an expression or a link from another element (e.g., a Time Series). It must increase monotonically (it cannot decrease with time). If the Count increases by more than one at any given time, the element will release multiple events simultaneously.

**Remaining time to event.** This event requires as input a Remaining time. This represents the time until the next event. This event type is designed to provide a straightforward way to insert events at specific times in the middle of a scheduled timestep. Every time the model is updated (e.g., every timestep), GoldSim recomputes Remaining time and reschedules the event. That is, this event is reset every timestep. Hence, for the event to occur, at some point the Remaining time value should be less than the scheduled timestep length (If the Remaining time was defined as a constant value that was greater than the timestep length, the event would never occur.) As an example, you could use this type of event to accurately represent when a vehicle reached a particular destination by defining the Remaining time as Distance_Remaining/Velocity.

Importance sampling of events is frequently necessary to evaluate the consequences of low-probability, high-consequence events (i.e., events that occur with a very low frequency, but have a significant impact on the system). Because the models for such systems are often complex (and hence need significant computer time to simulate), it can be difficult to use the conventional Monte Carlo approach to evaluate these low-probability, high-consequence events, as this may require excessive numbers of realizations.

To facilitate these type of analyses, GoldSim allows you to utilize an importance sampling algorithm to modify the conventional Monte Carlo approach so that the high-consequence, low-probability events are sampled with an enhanced frequency. That is, importance sampling serves to increase the rate of occurrence of the event. During the analysis of the results that are generated, the biasing effects of the importance sampling are reversed. The result is high-resolution development of the high-consequence, low-probability "tails" of the consequences (resulting from low-probability events), without paying a high computational price.

Importance sampling for Timed Events is specified by selecting the checkbox for Use Importance Sampling for this element. The algorithm that is used is discussed in detail in Appendix B.

Five points regarding importance sampling of events should be noted:

- Importance sampling can only be applied if the Occurrence Type is “Random time intervals (Poisson)” or “Stochastic element time intervals”.
- Importance sampling is only applied to the first occurrence of the event. The increased sampling frequency is not applied to subsequent occurrences within the same realization.
- Importance sampling of events should only be used for events that are rare. As used here, “rare” indicates an event that would not adequately be represented without enhanced sampling. As a general rule of thumb,
an event will be adequately sampled if the product of the number of realizations and the expected number of events over the course of a single realization is at least 10 (this would indicate that the event would be expected to occur at least 10 times if that many realizations were executed). For example, if the rate of occurrence of an event was once every thousand years, and the simulation was run for 10 years, one could expect 0.01 events per realization. If 100 realizations were run, the total expected number of events (over all realizations) would therefore be 1. This is a rare event and importance sampling should be applied (or more realizations should be run).

- There is a limit to the effectiveness of importance sampling for extremely rare events, such that in some cases, it may become necessary to increase the number of realizations in order to effectively represent the event. In particular, the degree of biasing for low probability events that GoldSim can provide is (at most) equal to the number of realizations. For example, if the rate of occurrence of an event was once every 100,000 hours, and the simulation was run for 10 hours, one could expect $1e^{-4}$ events per realization. If 100 realizations were run, the total expected number of events (over all realizations) would therefore be 0.01. This is a rare event and importance sampling should be applied. However, importance sampling could only increase the total number of events (over all realizations) by a factor of 100 (the number of realizations) to 1 (which would still provide inadequate representation). If 1000 realizations were run, the total expected number of events (over all realizations) without importance sampling would be 0.1, but importance sampling would improve it by a factor of 1000.

- You should use importance sampling sparingly (i.e., only for those elements that really need it). This is because, as stated above, the degree of biasing for low probability failures that GoldSim can provide is at most equal to the number of realizations, and the actual biasing provided decreases with the number of elements for which importance sampling is applied.

The importance sampling algorithm is discussed in detail in Appendix B.

With the exception of the Defined cumulative event count event type, you can specify a Maximum Number of Events to limit the total number of events that are generated.

Timed Event elements have the ability to influence when a model is updated. Typically, a model is updated at every “scheduled” timestep. That is, all the elements are computed at every timestep. A Timed Event, however, can force a model to be updated between “scheduled” timesteps.

For example, suppose that you have specified a 10 day timestep. If a Timed Event occurred at, say 23 days, GoldSim would insert an update (an “internal” event) between timesteps (i.e., at 23 days) in order to more accurately represent the event.

Read more: Understanding Timestepping in GoldSim (page 455).

Note: Timed Event elements with a Defined cumulative event count do not influence when the model is updated.
Generating Discrete Event Signals

Note: Timed Events are designed to interrupt the clock and insert a new update. If you choose to disable unscheduled updates, the events will be deferred to the next scheduled update.

Read more: Controlling Unscheduled Updates (page 470).

Timed Events have up to two inputs (a Rate, a Stochastic, a Count, or a Remaining time, along with the Maximum Number of Events).

Timed Events have two outputs: 1) the discrete event signal itself (the primary output); and 2) the EventCount (i.e., the cumulative number of events which have been emitted during the realization).

The discrete event signal output itself cannot be saved or viewed as a result.

Note: Element inputs and outputs are only shown in the browser if you choose to Show Element Subitems (accessed via the browser context menu by right-clicking in the browser).

The example model TimedTriggered.gsm in the General Examples/Events folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) contains an example of the use of Timed Event elements.

Triggered Event elements produce discrete event signals based on a specified trigger. You would use a Triggered Event if you needed to trigger a number of other elements (e.g., Discrete Change elements or Stochastics) with a single event.

The dialog for a Triggered Event is quite simple, containing only the Trigger button:

Triggered Event Elements
The Trigger… button in the Triggered Event dialog provides access to a standard Trigger dialog.

Read more: Understanding Event Triggering (page 355).

The inputs to a Triggered Event are the trigger(s) and any associated Precedence or Required Conditions.

Triggered Events have two outputs: 1) the discrete event signal itself (the primary output); and 2) the EventCount (i.e., the cumulative number of events that have been emitted during the realization).

These outputs can be seen in the browser view of the element:

The discrete event signal output itself cannot be saved or viewed as a result.

Note: Element inputs and outputs are only shown in the browser if you choose to Show Element Subitems (accessed via the browser context menu by right-clicking in the browser).

The example model TimedTriggered.gsm in the General Examples/Events folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) contains an example of the use of Triggered Event elements.

Triggered Events can have specified Resource Requirements.

Read more: Using Resources (page 839).

The Trigger dialog for a Triggered Event includes a Resources… button:

Specifying Resources for a Triggered Event
To define a Resource Requirement for a Triggered Event, press the Resources... button in the Trigger dialog for the element. The following dialog will be displayed:

You can add a Resource Requirement by pressing the Add button.

Read more: Interacting with Resources (page 848).

A Triggered Event interacts with the specified Resource Stores when triggered, and can only have two types of interactions (specified in the Usage column):

- **Spend (discrete):** A discrete quantity of the Resource is required in order to pull the trigger. If triggered with a Spend Requirement and the requested Resource quantity is not available, the trigger is held (it waits) until the Resource becomes available. If the Resource becomes available at a later time in the simulation, this creates a delay; if the Resource never becomes available, the trigger is never pulled.

- **Deposit (discrete):** A discrete quantity of the Resource is created and deposited with the Store when the element is triggered.

Resources are an advanced feature, and you should read the sections below before attempting to use them.

**Decision Elements**

Decision elements allow you to better represent decision or branching logic in your model.

Decision elements have either two or three outputs which represent discrete event signals. When the element is triggered, it emits a discrete event signal from one (and only one) of its outputs, depending on a set of user-specified conditions.

The dialog for a Decision element looks like this:
You first specify when the Decision is to be triggered via the Trigger… button. The Trigger… button in the Decision dialog provides access to a standard Trigger dialog.

**Read more:** [Understanding Event Triggering](#) (page 355).

When the element is triggered, it evaluates the Condition to test field to determine which output will emit a discrete event signal. By default, there are two discrete event signal outputs, which are named Case1 and Case3. The Condition to test field accepts any condition output or conditional expression.

The ID of output can be modified by the user to any string. (Note, however, that the same rules which apply for element names apply for these IDs.)

**Read more:** [Editing an Element’s Name, Description and Appearance](#) (page 78).

To understand how a Decision element works, consider the following example:
In this case, whenever the element is triggered, GoldSim evaluates the output “My\_Condition” (which must be a condition). If this condition is True, a discrete event signal is emitted from the Yes output of the element (in this case, named Decision1.\_Yes). Otherwise, a discrete event signal is emitted from the No output of the element (in this case, named Decision1.\_No).

You can add a second condition (and a third output) by clicking the checkbox to the left of the “else if” on the dialog.

In this case, whenever the element is triggered, if My\_Condition is True, a discrete event signal is emitted from the “Right” output of the element (in this case, named Decision1.\_Right). If My\_Condition is False, but A > B, then a discrete event signal is emitted from the “Left” output of the element (in this
case, named Decision1.Left). If neither of these conditions is True (i.e., My_Condition is False and A<= B), a discrete event signal is emitted from the “Center” output of the element (in this case, named Decision1.Center).

In addition to the two (or three) discrete event signal outputs associated with a Decision element (which you provide a name for), an additional output is produced called Last_Decision. This is a value which takes on one of four values: 0, 1, 2, or 3. Prior to the Decision being triggered, it takes on a value of 0. After a Decision element is triggered, it records which of the three discrete event signal outputs emitted the signal. These are always 1, 2 or 3. Note that even if there are only two outputs, the last event signal output (the “else”) is referred to as output 3.

**Note:** A Decision element could be triggered multiple times within a single update (e.g., a timestep), in which case it would output multiple events. Note, however, that these events would all be sent to the same output (since the Condition to test fields are only evaluated once each update).

The inputs to a Decision are the trigger(s) and any associated Precedence or Required Conditions, as well as the test conditions. Decisions have up to three discrete event signal outputs, along with the Last_Decision output.

The discrete event signal outputs themselves cannot be saved or viewed as a result.

**Note:** Element inputs and outputs are only shown in the browser if you choose to Show Element Subitems (accessed via the browser context menu by right-clicking in the browser).

The example model Decision.gsm in the General Examples/Events folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) contains an example of the use of Decision elements.

Random Choice elements allow you to represent event trees (processes consisting of specified random event sequences).

Random Choice elements have an unlimited number of specified outputs which represent discrete event signals. When the element is triggered, it emits a discrete event signal from one (and only one) of its outputs, depending on a set of user-specified probabilities.

The dialog for a Random Choice element looks like this:
You first specify when the Random Choice is to be triggered via the **Trigger...** button. The **Trigger...** button in the Decision dialog provides access to a standard Trigger dialog.

**Read more:** [Understanding Event Triggering](page 355).

You must then define a set of Probability/Event pairs. That is, you specify the name of an **Output Event** (a discrete event signal output) and a corresponding **Probability**.

By default, there are two Output Events, which are named Event1 and LastEvent. The **Output Event** name can be modified by the user to any string. (Note, however, that the same rules which apply for element names apply for these IDs.)

**Read more:** [Editing an Element’s Name, Description and Appearance](page 78).

You can define the **Probability** for all of the Events, except the last. The probability of the last event is always equal to 1 – (the sum of the other probabilities). The **Probability** can be specified as a number, links or expression, and can change with time. At all times, however, the specified values must sum to a value less than or equal to 1 (otherwise, a fatal error will be displayed when you run the simulation).

When the element is triggered, it randomly selects one of the **Output Events** based on the specified discrete probabilities, and emits a discrete event signal from that output.

To understand how a Random Choice element works, consider the following example:
In this case, whenever the element is triggered, GoldSim evaluates the Probabilities, and randomly selects an Output Event based on these Probabilities. In this particular case, there is a 10% chance that it will emit a discrete event signal from the Left output of the element (in this case, named RandomChoice1.Left), a 60% chance that it will emit a discrete event signal from the Right output of the element (in this case, named RandomChoice1.Right), and a 30% chance (1 – 0.1 – 0.6) that it will emit a discrete event signal from the Center output of the element (in this case, named RandomChoice1.Center).

In addition to the discrete event signal outputs associated with a Random Choice element (which you provide a name for), an additional output is produced called Last_Choice. This is a value which takes on an integer value of between 0 and n, where n is the number of Output Events defined. Prior to the Random Choice being triggered, it takes on a value of 0. After the Random Choice is triggered, it records which of the discrete event signal outputs emitted the signal (e.g., if the third event in the list was emitted, Last_Choice would be 3).

**Note:** A Random Choice element could be triggered multiple times within a single update (e.g., a timestep), in which case it would output multiple events. For each event, it would internally resample the Output Event.

Importance sampling is frequently necessary to evaluate the consequences of low-probability, high-consequence events (i.e., events that occur with a very low frequency, but have a significant impact on the system). Because the models for such systems are often complex (and hence need significant computer time to simulate), it can be difficult to use the conventional Monte Carlo approach to evaluate these low-probability, high-consequence events, as this may require excessive numbers of realizations.

To facilitate these type of analyses, GoldSim allows you to utilize an importance sampling algorithm to modify the conventional Monte Carlo
approach so that the high-consequence, low-probability events are sampled with an enhanced frequency. That is, importance sampling serves to increase the rate of occurrence of the event. During the analysis of the results that are generated, the biasing effects of the importance sampling are reversed. The result is high-resolution development of the high-consequence, low-probability "tails" of the consequences (resulting from low-probability events), without paying a high computational price.

Importance sampling for Random Choice elements is specified by selecting the checkbox for **Use Importance Sampling for this element**. The algorithm that is used is discussed in detail in Appendix B. For Random Choice elements, when importance sampling is applied, the alternatives are sorted by their probabilities, and the importance sampling serves to artificially bias (increase) the likelihood of the low probability alternatives (such that the least probable choice has the greatest magnification).

Three points regarding importance sampling for Random Choice elements should be noted:

- Importance sampling is only applied to the element the first time it is triggered. The increased sampling frequency for low probability alternatives is not applied to subsequent times the element is triggered in the realization.

- Importance sampling of Random Choice alternatives should only be used if at least one of the alternatives is rare. As used here, “rare” indicates that an alternative would not adequately be represented without enhanced sampling. As a general rule of thumb, an alternative will be adequately sampled if the product of the number of realizations and the probability of occurrence is at least 10 (this would indicate that the alternative would be expected to be selected at least 10 times if many realizations were executed). For example, if the probability of an alternative was 0.01 and 100 realizations were run, the total expected number of times the alternative would be selected (over all realizations) would therefore be 1. This is a rare alternative and importance sampling should be applied (or more realizations should be run).

- There is a limit to the effectiveness of importance sampling for extremely rare alternatives, such that in some cases, it may become necessary to increase the number of realizations in order to effectively represent the alternative. In particular, the degree of biasing for low probability alternatives that GoldSim can provide is (at most) equal to the number of realizations. For example, if probability of an alternative was 1e-4, and 100 realizations were run, the total expected number of times that alternative would be selected (over all realizations) would therefore be 0.01. This is a rare alternative and importance sampling should be applied. However, importance sampling could only increase the total number of occurrences (over all realizations) by a factor of 100 (the number of realizations) to 1 (which would still provide inadequate representation). If 1000 realizations were run, the total expected number of occurrences (over all realizations) without importance sampling would be 0.1, but importance sampling would improve it by a factor of 1000. As stated above, the degree of biasing for low probability alternatives that GoldSim can provide is at most equal to the number of realizations. The actual degree of biasing provided decreases with the number of elements for which importance sampling is applied (and the algorithm is discussed in detail in Appendix B).
The inputs to a Random Choice are the trigger(s) and any associated Precedence or Required Conditions, as well as the Probabilities. Random Choice elements have at least two discrete event signal outputs, along with the Last_Choice output.

The discrete event signal outputs themselves cannot be saved or viewed as a result.

**Note:** Element inputs and outputs are only shown in the browser if you choose to Show Element Subitems (accessed via the browser context menu by right-clicking in the browser).

The example model RandomChoice.gsm in the General Examples/Events folder of your GoldSim directory (accessed by selecting **File | Open Example...** from the main menu) contains an example of the use of Random Choice elements.

**Event Delay Elements**

Event Delay elements provide a mechanism for delaying a discrete event signal. Once an Event Delay receives a discrete event signal, it holds it for a specified time period, and then emits it.

The dialog for an Event Delay looks like this:
You first specify when the element is to be triggered via the Trigger... button in the Event Delay dialog. The Trigger... button provides access to a standard Trigger dialog.

**Read more:** Understanding Event Triggering (page 355).

Each time the element is triggered, it waits for a specified time, and then emits a discrete event signal.

The delay time is computed based on a specified Delay Type. There are four types:

- Defined Delay Time.
- Defined Delay Time + Erlang Dispersion
- Defined Delay Time + Std. Deviation
- Stochastic Delay Time Definition

The last three options allow the delay time to have a specified dispersion. That is, they allow you to simulate variability in when a signal is emitted once the element is triggered. This is equivalent to saying that there is a distribution of actual delay times around some mean, and whenever the element is triggered, the delay time for that event is sampled from a distribution.

The first three options require a specified Delay Time (or a Mean Time if dispersion is specified). The Delay Time, and Mean Time must have dimensions of time and generally should be positive.

The fourth option requires a Stochastic that represents a distribution of Delay Times. The Stochastic element that is specified represents the distribution of the delay time. As such, it must have dimensions of time. The Distribution output of the Stochastic must be referenced:

*\[\text{Delay Definition}\]

- **Delay Type:** Stochastic Delay Time Definition
- **Stochastic:** DelayTime.Distribution

**Note:** If the specified delay time is equal to zero, the event is emitted immediately (i.e., on the same update) without any delay. A negative delay time will result in a fatal error.

**Note:** The Use conveyer-belt approach box only impacts the behavior of the Delay if the delay time changes with time.

**Read more:** Event Delays with Time-Variable Delay Times (page 381).

You can optionally specify that an Event Delay requires one or more Resources in order to process the signal.

**Read more:** Specifying Resources for an Event Delay (page 382).

Event Delay elements have the ability to influence when a model is updated. Typically, a model is updated at every “scheduled” timestep. That is, all the elements are computed at every timestep. However, an Event Delay can force a model to be updated between “scheduled” timesteps.
For example, suppose that you have specified a 10 day timestep. If an Event Delay receives a signal at 20 days, and has a 3 day delay time, GoldSim will insert an update (an “internal” event) between timesteps (i.e., at 23 days) in order to more accurately represent the event.

*Read more:* [Understanding Timestepping in GoldSim](page 455).

---

**Warning:** Event Delays are designed to interrupt the clock and insert a new update when the event is released. If you choose to disable unscheduled updates, the event in the Delay will be deferred to the next scheduled update (and hence the actual Delay Time will be longer than specified).

*Read more:* [Controlling Unscheduled Updates](page 470).

Event Delays have at least five, and as many as six outputs:

1. the discrete event signal itself (the primary output);
2. the cumulative number of discrete signals emitted (released) by the Delay (EventCount);
3. the number of discrete signals currently in transit within the Delay (Num_in_Transit);
4. the number of discrete signals currently in the queue (Num_in_Queue), which is only present if a capacity or a Resource Requirement is specified;
5. the mean time that emitted signals have spent in the delay (Mean_Time); and
6. the current service time (Current_Service_Time). This represents the expected time it will take a new item (event) to transit the system. If there is no queue (no capacity or Resource requirements specified), or if no events have transited the system, then the current service time is simply reported as -1 (indicating no data). If there is a queue and at least one event has transited the system, the current service time is approximated as the time it took the most recent item to transit the system.

Note that the discrete event signal output itself cannot be saved or viewed as a result.

The example model EventDelay.gsm in the General Examples/Events folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) contains an example of the use of Event Delay elements.

The simplest (and most common) use of an Event Delay element is to represent the delay of a non-dispersed signal. To simulate such a system, the Delay Type should be set to “Defined Delay Time”. In this case, after being triggered, the element “holds” the signal for a time period equal to the specified Delay Time, and after this delay, emits a discrete event signal.

In some cases, there may be some variability in when a signal is emitted once the element is triggered. This is equivalent to saying that there is a distribution of actual delay times around some mean, and whenever the element is triggered, the delay time for that event is sampled from a distribution.

If we conceptualize the Delay as a conveyer belt for the signal, another way to view event dispersion is that as the signal moves along the belt, it slips randomly...
forward or backward on the belt, with the amount of movement proportional to the degree of dispersion.

You can specify such dispersion in the delay time in three different ways:

- By specifying the dispersion in terms of an Erlang dispersion factor;
- By specifying the dispersion in terms of a standard deviation; or
- By specifying a custom distribution of delay times (by referencing a Stochastic element).

If “Defined Delay Time + Erlang Dispersion” is selected, you must enter an **Erlang n-value**, which is a dimensionless value greater than or equal to 1. As n increases, the degree of dispersion decreases. As n goes to infinity, the dispersion goes to zero. The maximum amount of dispersion allowed is represented by n = 1.

If “Defined Delay Time + Std. Deviation” is selected, you must enter a **Std. Deviation**, which is a value with dimensions of time. The value must be greater than or equal to zero and less than or equal to the **Mean Time**. As the **Std. Deviation** decreases, the degree of dispersion decreases. When the **Std. Deviation** goes to zero, the dispersion goes to zero. The maximum amount of dispersion allowed is represented by **Std. Deviation = Delay Time**.

The Erlang n and the Std. Deviation are related by the following equation:

\[
n = \left( \frac{\text{Delay Time}}{\text{Std. Deviation}} \right)^2
\]

If the signal is dispersed, for every event, the Delay Time for that event is sampled from the following distribution:

\[
f(t) = \frac{t^{n-1} e^{-t/\beta}}{\beta^n \Gamma(n)}
\]

where:

- n is the Erlang value (specified by the user);
- \( \beta = D/n \);
- D is the mean delay time; and
- \( \Gamma \) is the gamma function (not the Gamma distribution).

f(t) is the gamma probability distribution, which is equivalent to (and a generalization of) the Erlang distribution that is frequently used in simulation models.

The gamma distribution represents the time until the occurrence of n sequential Poisson-process events. Each event’s random time is represented by an exponential distribution with mean D/n. The gamma distribution does not require n to be an integer. Note that for n=1, the distribution is exponential, and for increasing values of n it becomes less skewed, approaching normality for large n.

The “Stochastic Delay Time Definition” option provides a way to specify a custom distribution for the delay time (rather than the gamma used for the Erlang and Std. Deviation options). If this option is selected, you must specify a **Stochastic** element. The Stochastic element that is specified represents the
distribution of the delay time. As such, it must have dimensions of time. The Distribution output of the Stochastic must be referenced:

Note: When dispersion is specified for a Delay, each event which triggers the element is “assigned” an actual delay time by sampling from the distribution presented above. As a result, when Event Delays are dispersed, the events will not necessarily be “released” in the order that they were received.

**Event Delays with Time-Variable Delay Times**

In some cases, the delay time for your Event Delay may be specified to change as a function of time. You would do this by directly specifying the **Delay Time** or **Mean Time** as varying as a function of time.

Whenever the **Delay Time** or **Mean Time** are specified to vary with time in a simulation, you have two options as to how GoldSim can represent this.

If the **Use conveyer-belt approach** box is checked, GoldSim will treat the delay as if it is a conveyer belt. In particular, if the **Delay Time** or **Mean Time** is specified to become shorter (or longer) during the simulation (e.g., by defining it as a function of time), it can be imagined that the speed at which the belt moves has simply been increased (or decreased), and all events that are in the Delay at the time of the change start to move faster (or slower) by a common factor (the ratio of the old **Delay** or **Mean Time** to the new one).

For example, if the Event Delay was triggered at 10 days while the **Delay Time** was equal to 1000 days, and again at 15 days at which time the **Delay Time** was equal to 1 day, both signals would effectively be emitted at 16 days. That is, at the time that the **Delay Time** decreased, the first event would not have traversed a significant distance along the conveyer, and therefore it would be emitted from the conveyer just slightly in front of the second event.

If the **Use conveyer-belt approach** box is cleared, each event is assigned an effective delay time when it enters the Delay. Once an event enters the Delay, its delay time is not impacted at all if the **Delay Time** or **Mean Time** is subsequently specified to become shorter (or longer) during the simulation. Referring to the example above, if the **Use conveyer-belt approach** box was cleared, then the first event would be emitted at 1010 days, while the second event would be emitted at 16 days.

Note: When using the “Stochastic Delay Time Definition” option, the Delay can never act as a conveyer (the **Use conveyer-belt approach** box is cleared and grayed out). Every event is assigned an actual delay time by sampling from the Stochastic. If the Stochastic is modified during a simulation, it does not affect events that have already been assigned a delay time.

**Specifying Capacities and Modeling Queues for an Event Delay**

By checking the **Maximum number of signals simultaneously processed** checkbox within the Event Delay dialog, you can specify a capacity for the Delay in the input field directly below the checkbox.

For example, if you specify this value as 3, the element can only have three signals in transit at any given time. Any other signals that it receives are placed in a queue. Signals in the queue must wait for one of the signals in transit to be
emitted before they can begin to traverse the delay. GoldSim keeps track of the number of signals in transit (Num_in_Transit), the number of signals in the queue (Num_in_Queue), the mean time that all emitted signals spent in the Delay (Mean_Time), and the time it will take any newly arriving items to transit the system (Current_Service_Time). These latter two outputs account for both the time spent in the queue and the time spent in transit.

These features allow you to model things such as lines at the grocery store, calls at a call center, traffic on a road, and a wide variety of other real-world queues.

![Note:](image)

Note: The specified capacity can be a number, a link or an expression. It must be dimensionless and non-negative. GoldSim automatically truncates the value to an integer.

### Specifying Resources for an Event Delay

Event Delays can have specified Resource Requirements.

**Read more:** Using Resources (page 839).

To define a Resource Requirement for an Event Delay, press the Resources... button on the main dialog for the element. The following dialog will be displayed:

![Event Delay](image)

You can add a Resource Requirement by pressing the Add button.

**Read more:** Interacting with Resources (page 848).

An Event Delay interacts with the specified Resource Stores when a Signal arrives in the Delay (via the Trigger dialog), and can only have three types of interactions (specified in the Usage column):

- **Spend (discrete):** A discrete quantity of the Resource is required in order to begin to process the Signal. If the requested Resource quantity is not available, the Signal is added to the element’s queue, where it waits for the Resource to become available.

- **Borrow (discrete):** A discrete quantity of the Resource is required in order to begin to process the Signal. If the requested Resource quantity is not available, the Signal is added to the element’s queue, where it waits for the Resource to become available. Once the Signal leaves the Delay, the borrowed quantity is returned to the Resource Store.

- **Deposit (discrete):** A discrete quantity of the Resource is created and deposited with the Store when the element begins to process the Signal.

Resources are an advanced feature, and you should read the sections below before attempting to use them.
The browser view of an Event Delay includes all of its inputs (e.g., the Delay Time, Dispersion, and Trigger) and up to six outputs:

The discrete event signal output itself cannot be saved or viewed as a result.

**Note:** Element inputs and outputs are only shown in the browser if you choose to **Show Element Subitems** (accessed via the browser context menu by right-clicking in the browser).

### Responding to Events

Elements such as Timed Events and Decisions can emit (and in some cases, receive) discrete change signals. Ultimately, however, discrete event modeling must involve discretely impacting an element in some manner in response to the occurrence of the event. There are five elements that, when triggered, respond to an event:

- Discrete Changes;
- Statuses;
- Milestones;
- Stochastics; and
- Interrupts.

The manner in which these elements respond to events is described below.

The Discrete Change element is triggered by an event, and responds by emitting a discrete change signal. The triggering event can be a discrete event signal or another type of event (e.g., a condition, such as X becoming greater than Y).

The discrete change signal output by the element can only be used by a number of elements. In particular, it can be delayed by a Discrete Change Delay element; and it can instantaneously impact the value of an Integrator, a Reservoir or a Pool. It can also act as an input to some elements in GoldSim's specialized modules (e.g., the Financial Module).

**Read more:** [Propagating Discrete Signals Between Elements](page 353).

The dialog for a Discrete Change element looks like this:
You first specify when the element is to be triggered via the Trigger... button in the Discrete Change dialog. The Trigger... button provides access to a standard Trigger dialog.

Read more: Understanding Event Triggering (page 355).

Once the element is triggered, it emits a discrete change signal. A discrete change signal contains information regarding the response to an event and consisting of two pieces of information: a Value (e.g., 10 dollars) and an Instruction (e.g., Add).

The Value can be a number (e.g., 10 m3) or a link. The required order (scalar, vector or matrix) of the Value is specified in a dialog accessed via the Type... button. The required dimensions of the Value are specified in the Display Units field. By default, the Value is a dimensionless scalar.

The Instruction drop-list contains three choices: "Add", "Replace" and “Push”. This instructs the element(s) receiving the discrete change signal (e.g., an Integrator, a Reservoir or a Pool) how to act upon the value being received.

While Add and Replace have obvious implications for how they are handled by the receiving element (and are the most commonly used), a Push instruction is more complex (and used less often), and is intended specifically to facilitate modeling of aging chains.

Read more: Modeling Aging Chains (page 810).

Note: A discrete change signal can be used as an argument to an On Event triggering event (in the same manner that a discrete event signal can).

Read more: Specifying Triggering Events (page 356).

Discrete Change elements have two outputs: 1) the discrete change signal itself (the primary output); and 2) the EventCount (i.e., the cumulative number of discrete change signals that the element has emitted during the realization.

These outputs can be seen in the browser view of the element:
The discrete change signal output itself cannot be saved or viewed as a result.

**Note:** Element inputs and outputs are only shown in the browser if you choose to **Show Element Subitems** (accessed via the browser context menu by right-clicking in the browser).

The example model DiscreteChange.gsm in the General Examples/Events folder of your GoldSim directory (accessed by selecting **File | Open Example...** from the main menu) contains an example of the use of Discrete Change elements.

Integrator elements can be instantaneously impacted by a discrete change signal.

**Read more:** [Integrator Elements](page 228).

The Integrator dialog has a special input field for accepting a discrete change signal:

Discrete changes to the Integrator are entered in the **Discrete Change** field, or by clicking the **Multiple Discrete Signals** button.

The **Discrete Change** input only accepts discrete change signals. The dimensions and order of the discrete change signals must be the same as those of the Integrator.

When an Integrator receives a discrete change signal with an Add instruction, it instantaneously adds the value of the signal to the Current Value of the Integrator. When an Integrator receives a discrete change signal with a Replace instruction, it instantaneously replaces the Current Value of the Integrator with the value of the signal.
An Integrator can also accept a discrete change signal with a Push instruction, which facilitates the modeling of aging chains.

**Read more:** [Modeling Aging Chains](#) (page 810).

An Integrator can accept multiple discrete changes. This is indicated in the input field by separating the individual discrete change signals by semi-colons (e.g., Change1; Change2; Change3).

You can also specify the multiple discrete changes using the Multiple Discrete Signals button, which displays a table listing the multiple discrete change signals:

A simple (and fun) application of Discrete Change elements, simulating a bouncing ball (Bounce.gsm), is included in the General Examples folder of your GoldSim directory (accessed by selecting **File | Open Example...** from the main menu).

Reservoir elements can be instantaneously impacted by a discrete change signal. Moreover, a Reservoir can also produce (emit) discrete change signals.

**Read more:** [Reservoir Elements](#) (page 235).

The Reservoir dialog has special input fields for accepting a discrete change signal:
Discrete changes to the Reservoir are entered in the Discrete Change fields (Additions or Withdrawal Requests).

If the checkbox to the left of Discrete Change under Additions in the Reservoir dialog is checked, then:

- the input field is activated; and
- if the Reservoir has an Upper Bound, a new output called Discrete_Overflow is created on the element.

The Discrete Change input under Additions accepts discrete change signals which have a positive value Add instruction. It also accepts discrete change signals which have a Replace instruction. The dimensions and order of the discrete change signal must be the same as those of the Reservoir.

When a Reservoir receives a Discrete Change Addition with an Add instruction, it instantaneously adds the value of the signal to the Value of the Reservoir. When a Reservoir receives a Discrete Change Addition with a Replace instruction, it instantaneously replaces the Value of the Reservoir with the value of the signal.

If the Reservoir has an Upper Bound and this is exceeded by the Discrete Change Addition, the Value is fixed at the Upper Bound, and the Reservoir produces (emits) a discrete change signal (Discrete_Overflow) which represents the amount of the Discrete Change Addition which "overflowed" the Reservoir.

Read more: Defining Upper and Lower Bounds for a Reservoir (page 240).

Note: If you have specified an Upper or Lower Bound, and choose to Replace the current value of a Reservoir, the replacement value must be within the bounds (or GoldSim will display a fatal error message).
If the checkbox to the left of **Discrete Change** under **Withdrawal Requests** is checked, then

- the input field is activated; and
- a new output called Discrete_Withdrawals is created on the element.

The **Discrete Change** input under **Withdrawal Requests** only accepts discrete change signals which have a positive value and an Add instruction. The dimensions and order of the discrete change signal must be the same as those of the Reservoir.

When a Reservoir receives a Discrete Change Withdrawal Request, it instantaneously subtracts the value of the signal from Value of the Reservoir. In addition, a discrete change signal (Discrete_Withdrawals) which echoes the Discrete Change Withdrawal Request that was removed from the Reservoir is automatically emitted.

If the Reservoir has a **Lower Bound** and the Discrete Change Withdrawal Request forces the Reservoir to go below this value, the Value is fixed at the **Lower Bound**, and the Reservoir produces (emits) a discrete change signal (Discrete_Withdrawals) which represents the actual Discrete Change Withdrawal that was removed from the Reservoir (i.e., the difference between the Value prior to the withdrawal and the **Lower Bound**).

**Read more:** [Defining Upper and Lower Bounds for a Reservoir](page 240).

A Reservoir can accept multiple Discrete Changes (both Additions and Withdrawal Requests). This is indicated in the input field by separating the individual discrete change signals by semi-colons (e.g., Change1; Change2; Change3).

You can also specify the multiple discrete changes using the Multiple Discrete Signals button, which displays a table listing the multiple discrete change signals:

Pool elements can be instantaneously impacted by a discrete change signal. Moreover, a Pool can also produce (emit) discrete change signals.

**Read more:** [Pool Elements](page 252).

The Definition tab of the Pool dialog has special input fields for accepting a discrete change signal:
Discrete changes to the Pool are entered in the Discrete Change fields (Additions or Withdrawals).

If the checkbox to the left of Additions in the Pool dialog is checked, then:

- the input field is activated; and
- if the Pool has an Upper Bound, a new output called Discrete_Overflow is created on the element.

The Additions input accepts discrete change signals which have a positive value Add instruction. It also accepts discrete change signals which have a Replace instruction. The dimensions of the discrete change signal must be the same as those of the Quantity Units.

When a Pool receives a Discrete Change Addition with an Add instruction, it instantaneously adds the value of the signal to the Quantity of the Pool. When a Pool receives a Discrete Change Addition with a Replace instruction, it instantaneously replaces the Quantity of the Pool with the value of the signal.

If the Pool has an Upper Bound and this is exceeded by the Discrete Change Addition, the Quantity is fixed at the Upper Bound, and the Pool produces (emits) a discrete change signal (Discrete_Overflow) which represents the amount of the Discrete Change Addition which "overflowed" the Pool.

Read more: Defining Upper and Lower Bounds for a Pool (page 258).

Note: If you have specified an Upper or Lower Bound, and choose to Replace the current value of a Pool, the replacement value must be within the bounds (or GoldSim will display a fatal error message).

If the checkbox to the left of Withdrawals is checked, then

- the input field is activated; and
- a new output called Discrete_Withdrawals is created on the element.
The *Withdrawals* input only accepts discrete change signals which have a positive value and an Add instruction. The dimensions of the discrete change signal must be the same as those of the *Quantity Units*.

When a Pool receives a Discrete Change Withdrawal, it instantaneously subtracts the value of the signal from Quantity of the Pool. In addition, a discrete change signal (Discrete-Withdrawals) which echoes the Discrete Change Withdrawal Request that was removed from the Reservoir is automatically emitted.

If the Pool has a *Lower Bound* and the Discrete Change Withdrawal forces the Pool to go below this value, the Quantity is fixed at the *Lower Bound*, and the Pool produces (emits) a discrete change signal (Discrete-Withdrawals) which represents the actual Discrete Change Withdrawal that was removed from the Pool (i.e., the difference between the Quantity prior to the withdrawal and the *Lower Bound*).

*Read more:* [Defining Upper and Lower Bounds for a Pool](page 258).

A Pool can accept multiple Discrete Changes (both Additions and Withdrawals). This is indicated in the input field by separating the individual discrete change signals by semi-colons (e.g., Change1; Change2; Change3).

You can also specify the multiple discrete changes using the Multiple Discrete Signals button, which displays a table listing the multiple discrete change signals:

![Multiple Discrete Signals button](image)

**Delaying a Discrete Change Signal**

Discrete Change Delay elements provide a mechanism for delaying a discrete change signal. Once a Discrete Change Delay receives a discrete change signal, it holds it for a specified time period, and then emits it.

The dialog for a Discrete Change Delay looks like this:
Within the Discrete Change Delay dialog, you must first specify which discrete change signal(s) you wish to delay. The **Discrete Input Signals** field accepts only discrete change signals. The discrete change signal inputs to a Discrete Change Delay must all have the same attributes (order and dimensions) as the output. The required order (scalar, vector or matrix) of the output signal is specified in a dialog accessed via the **Type...** button. The required dimensions of the signal are specified in the **Display Units** field. By default, the signal is a dimensionless scalar.

A Discrete Change Delay can accept multiple discrete change signals. This is indicated in the input field by separating the individual discrete change signals by semi-colons (e.g., Change1; Change2; Change3).

You can also specify the multiple discrete changes using the Multiple Discrete Signals button, which displays a table listing the multiple discrete change signals:
After defining the signal(s) to be delayed, you then define the delay itself. The delay time is computed based on a specified **Delay Type**. There are three types:

- Defined Delay Time.
- Defined Delay Time + Erlang Dispersion
- Defined Delay Time + Std. Deviation

The last two options allow the delay time to have a specified dispersion. That is, they allow you to simulate variability in when a signal is emitted once the element is triggered. This is equivalent to saying that there is a distribution of actual delay times around some mean, and whenever the element is triggered, the delay time for that event is sampled from a distribution.

All three options require a specified **Delay Time** (or a **Mean Time** if dispersion is specified). The **Delay Time** and the **Mean Time** must have dimensions of time and generally should be positive.

**Note:** If the specified delay time is equal to zero, the signal is emitted immediately (i.e., on the same update) without any delay. A negative delay time will result in a fatal error.

**Note:** The **Use conveyer-belt approach** box only impacts the behavior of the Delay if the delay time changes with time.

**Read more:** Discrete Change Delays with Time-Variable Delay Times (page 394).

You can optionally specify that a Discrete Change Delay requires one or more **Resources** in order to process the signal.

**Read more:** Specifying Resources for a Discrete Change Delay (page 396).

Discrete Change Delay elements have the ability to influence when a model is updated. Typically, a model is updated at every “scheduled” timestep. That is, all the elements are computed at every timestep. However, an Event Delay can force a model to be updated between “scheduled” timesteps.

For example, suppose that you have specified a 10 day timestep. If a Discrete Change Delay receives a signal at 20 days, and has a 3 day delay time, GoldSim will insert an update (an “internal” event) between timesteps (i.e., at 23 days) in order to more accurately represent the event.

**Read more:** Understanding Timestepping in GoldSim (page 455).

**Warning:** Discrete Change Delays are designed to interrupt the clock and insert a new update when the discrete change is released. If you choose to disable unscheduled updates, the discrete change in the Delay will be deferred to the next scheduled update (and hence the actual Delay Time will be longer than specified).

**Read more:** Controlling Unscheduled Updates (page 470).

Discrete Change Delays have at least three, and as many as four outputs:

1. the discrete change signal itself (the primary output);
2. the cumulative number of discrete change signals emitted (released) by the Delay (EventCount);
3. the number of discrete change signals currently in transit within the Delay (Num_in_Transit);
4. the number of discrete change signals currently in the queue (Num_in_Queue), which is only present if a capacity or a Resource Requirement is specified;

Note that the discrete change signal output itself cannot be saved or viewed as a result.

The browser view of the Discrete Change Delay element looks like this:

Note: Element inputs and outputs are only shown in the browser if you choose to Show Element Subitems (accessed via the browser context menu by right-clicking in the browser).

The example model DiscreteChange.gsm in the General Examples/Events folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) contains examples of the use of Discrete Change Delay elements.

Modeling Discrete Change Delays without Dispersion

The simplest (and most common) use of a Discrete Change Delay element is to represent the delay of a non-dispersed signal.

To simulate such a system, the Delay Type should be set to “Defined Delay Time”. In this case, after being triggered, the element “holds” the signal for a time period equal to the specified Delay Time, and after this delay, emits a discrete change signal.

In some cases, when you are delaying a discrete change signal in a Discrete Change Delay element, there may be some variability in delay times for signals. This is equivalent to saying that there is a distribution of actual delay times around some mean, and whenever the element receives a signal, the delay time for that signal is sampled from a distribution.

If we conceptualize the Delay as a conveyer belt for the signal, another way to view signal dispersion is that as the signal moves along the belt, it slips randomly forward or backward on the belt, with the amount of movement proportional to the degree of dispersion.

You can specify such dispersion in the delay time in two different ways:

- By specifying the dispersion in terms of an Erlang dispersion factor; or
- By specifying the dispersion in terms of a standard deviation.

Modeling Discrete Change Delays with Dispersion
If “Defined Delay Time + Erlang Dispersion” is selected, you must enter an **Erlang n-value**, which is a dimensionless value greater than or equal to 1. As n increases, the degree of dispersion decreases. As n goes to infinity, the dispersion goes to zero. The maximum amount of dispersion allowed is represented by n = 1, which corresponds to an exponential distribution of delay times.

If “Defined Delay Time + Std. Deviation” is selected, you must enter a **Std. Deviation**, which is a value with dimensions of time. The value must be greater than or equal to zero and less than or equal to the **Mean Time**. As the **Std. Deviation** decreases, the degree of dispersion decreases. When the **Std. Deviation** goes to zero, the dispersion goes to zero. The maximum amount of dispersion allowed is represented by **Std. Deviation = Delay Time**.

The Erlang n and the Std. Deviation are related by the following equation:

\[ n = \left( \frac{\text{Delay Time}}{\text{Std. Deviation}} \right)^2 \]

If the signal is dispersed, for every signal received, the Delay Time for that signal is sampled from the following distribution:

\[ f(t) = \frac{t^{n-1}e^{-t/\beta}}{\beta^n \Gamma(n)} \]

where:

- n is the Erlang value (specified by the user);
- \( \beta = D/n; \)
- D is the mean delay time; and
- \( \Gamma \) is the gamma function (not the Gamma distribution).

f(t) is the gamma probability distribution, which is equivalent to (and a generalization of) the Erlang distribution that is frequently used in simulation models.

The gamma distribution represents the time until the occurrence of n sequential Poisson-process events. Each event’s random time is represented by an exponential distribution with mean D/n. The gamma distribution does not require n to be an integer. Note that for n=1, the distribution is exponential, and for increasing values of n it becomes less skewed, approaching normality for large n.

---

**Note:** When dispersion is specified for a delay time, each signal received by the element is “assigned” an actual delay time by sampling from the distribution presented above. As a result, when signals in a Discrete Change Delay are dispersed, the signals will not necessarily be "released" in the order that they were received.

---

**Discrete Change Delays with Time-Variable Delay Times**

In some cases, the delay time for your Discrete Change Delay may be specified to change as a function of time. You would do this by directly specifying the **Delay Time** or **Mean Time** as varying as a function of time.

Whenever the **Delay Time** or **Mean Time** are specified to vary with time in a simulation, you have two options as to how GoldSim can represent this.
If the **Use conveyer-belt approach** box is checked, GoldSim will treat the delay as if it is a conveyer belt. In particular, if the **Delay Time** or **Mean Time** is specified to become shorter (or longer) during the simulation (e.g., by defining it as a function of time), it can be imagined that the speed at which the belt moves has simply been increased (or decreased), and all items that are in the Delay at the time of the change start to move faster (or slower) by a common factor (the ratio of the old **Delay** or **Mean Time** to the new one).

For example, if the Discrete Change Delay was triggered at 10 days while the **Delay Time** was equal to 1000 days, and again at 15 days at which time the **Delay Time** was equal to 1 day, both signals would effectively be emitted at 16 days. That is, at the time that the **Delay Time** decreased, the first item would not have traversed a significant distance along the conveyer, and therefore it would be emitted from the conveyer just slightly in front of the second item.

If the **Use conveyer-belt approach** box is cleared, each item is assigned an effective delay time when it enters the Delay. Once an item enters the Delay, its delay time is not impacted at all if the **Delay Time** or **Mean Time** is subsequently specified to become shorter (or longer) during the simulation. Referring to the example above, if the **Use conveyer-belt approach** box was cleared, then the first item would be emitted at 1010 days, while the second item would be emitted at 16 days.

**Note:** The **Use conveyer-belt approach** option is not available if you have referenced the discrete change signal Value directly when specifying the delay time.

**Referencing the Discrete Change Value to Determine the Delay Time**

In some cases, the delay time for a Discrete Change Delay may be a function of the Value that is associated with the Discrete Change signal being processed. For example, perhaps the Discrete Change Delay represented a manufacturing or treatment process, the Value of the signal represented the size of the item to be processed, and the processing time was a function of the size.

This can be simulated by referencing "~DC_Value", which is a locally available property within the Discrete Change Delay dialog:

![Delay Definition](image)

**Read more:** [Understanding Locally Available Properties](page 808).

In this example, if the Value of the incoming signal is less than 10kg, it takes 30 minutes to process; otherwise it takes 60 minutes.

**Note:** The ~DC_Value property is not available if the **Use conveyer-belt approach** option is selected. This is because when using the conveyer-belt approach, *all* signals in the delay are impacted when the Delay Time changes, and in almost all cases, it would therefore not make sense to adjust the Delay Time based on the attribute of one particular signal.

**Read more:** [Discrete Change Delays with Time-Variable Delay Times](page 394).
Note that the "~DC_Value" locally available property can be referenced in any field in the Discrete Change Delay element, including those associated with specifying Resource requirements.

**Read more:** [Specifying Resources for a Discrete Change Delay](#) (page 396).

The ability to reference the discrete change signal Value within a Discrete Change Delay, coupled with the ability to reference the discrete change signal Value within a Splitter, enables a wide range of discrete event processes to be simulated within GoldSim.

**Read more:** [Using Splitter Elements to Route Discrete Changes Based on Their Value](#) (page 397).

By checking the Maximum number of signals simultaneously processed checkbox within the Discrete Change Delay dialog, you can specify a capacity for the Delay in the input field directly below the checkbox.

For example, if you specify this value as 3, the element can only have three signals in transit at any given time. Any other signals that it receives are placed in a queue. Signals in the queue must wait for one of the signals in transit to be emitted before they can begin to traverse the delay. GoldSim keeps track of the number of signals in transit (Num_in_Transit) and the number of signals in the queue (Num_in_Queue).

**Note:** The specified capacity can be a number, a link or an expression. It must be dimensionless and non-negative. GoldSim automatically truncates the value to an integer.

### Specifying Capacities and Modeling Queues for a Discrete Change Delay

Discrete Change Delays can have specified Resource Requirements.

**Read more:** [Using Resources](#) (page 839).

To define one or more Resource Requirements for a Discrete Change Delay, press the Resources... button on the main dialog for the element. The following dialog will be displayed:

![Resource Requirements Dialog](image)

You can add a Resource Requirement by pressing the Add button.

**Read more:** [Interacting with Resources](#) (page 848).

A Discrete Change Delay interacts with the specified Resource Stores when a discrete change signal arrives in the Delay, and can only have three types of interactions (specified in the Usage column):
• **Spend (discrete):** A discrete quantity of the Resource is required in order to begin to process the Signal. If the requested Resource quantity is not available, the Signal is added to the element’s queue, where it waits for the Resource to become available.

• **Borrow (discrete):** A discrete quantity of the Resource is required in order to begin to process the Signal. If the requested Resource quantity is not available, the Signal is added to the element’s queue, where it waits for the Resource to become available. Once the Signal leaves the Delay, the borrowed quantity is returned to the Resource Store.

• **Deposit (discrete):** A discrete quantity of the Resource is created and deposited with the Store when the element begins to process the Signal.

---

**Note:** The Quantity/Rate field can reference the locally available property ~DC_Value (the Value that is associated with the Discrete Change signal being processed). This allows you to define a Resource Requirement that is a function of this value.

---

**Read more:** Referencing the Discrete Change Value to Determine the Delay Time (page 395).

Resources are an advanced feature, and you should read the sections below before attempting to use them.

In some situations, you may want to route a discrete change signal based on the Value it is carrying. For example, based on the Value associated with the discrete change signal, you may want to route the item to different processing streams. This can be accomplished by using a Splitter to route the discrete change signal based on its Value. This is possible because 1) a Splitter can accept (and route) discrete change signals; and 2) the specified fraction (or amount) for a Splitter output can reference the value of the signal using the ~Amount local property.

**Read more:** Splitter Elements (page 281).

To illustrate this, it is instructive to consider a simple example. This example can be found in the file Splitter.gsm in the General Examples folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

In this example, a discrete change signal with an “Add” instruction is generated. If the value is less than 100, we want to send it to one set of processes; if it is greater than 100, we want to send it to a different set of processes. The Splitter is used to route it in the appropriate direction:
The Splitter accomplishes this by referencing "~Amount", which is a locally available property within the Splitter dialog:

**Note:** If you only wanted to adjust the delay time for a single process, you could do so without using the Splitter at all by defining a single Discrete Change Delay and referencing the ~DC_Value. The example above is simply meant to illustrate how you could split a signal into one of several processing directions (which might each have many subsequent processing steps).

**Read more:** [Understanding Locally Available Properties](#) (page 808).

This technique works because a Splitter does not propagate a discrete change signal with a value of zero and an "Add" instruction ("Replace" and "Push" instructions with zero values would be propagated). Hence, given the logic specified above, only one signal is propagated whenever a discrete change signal is received by the Splitter.

**Status Elements**

Status elements allow you to monitor the status of something that is changed instantaneously (i.e., is triggered) by events. The single output of a Status element is a condition (True or False). The element has an initial condition (True or False) and two triggers: one that sets the Status to True, and another that sets the Status to False.

You would use a Status element to track the status or state of processes or objects in your model. For example, if you were modeling a material handling facility, you could use a Status element to signify whether or not a particular storage area could accept any more material (i.e., is it full?). You might then reference the output of the Status element in a Selector element or a Decision element to determine the flow of material in your model.

**Read more:** [Referencing the Discrete Change Value to Determine the Delay Time](#) (page 395).
One of the most powerful uses of a Status element is to represent a deadband when modeling an active feedback control system.

**Read more:** [Simulating a Deadband Using a Status Element](#) (page 400).

The dialog for a Status element looks like this:

![Status element dialog](image)

Within the Status dialog, you should first specify the **Initial Condition** for the element (True or False). This can be entered as a condition (e.g., False) or a conditional expression. By default, it is False.

**Note:** The **Initial Condition** cannot be a function of Time.

You must then specify the triggering event(s) which cause the element to be set to True and False. These are both defined via **Trigger**... buttons, which provide access to standard Trigger dialogs.

**Read more:** [Understanding Event Triggering](#) (page 355).

**Note:** You don’t have to specify both the True and the False trigger. However, if you do not specify the trigger that represents the opposite of the **Initial Condition**, GoldSim will issue a warning when you close the dialog.

To understand how a Status element works, consider the following simple example. Assume that a Status element defaults to False. A triggering event is defined which sets the Status to True whenever an event occurs (regularly, every 10 days). This event also triggers an Event Delay (which has a delay of 2 days). The output of the Event Delay then sets the Status to False (2 days after it was set to True). The structure of this model looks like this:
Simulating a Deadband Using a Status Element

In many engineered (and some social and organizational) systems, there is active feedback control designed directly into the system to make it behave in a

Note: Element inputs and outputs are only shown in the browser if you choose to Show Element Subitems (accessed via the browser context menu by right-clicking in the browser).

The example model StatusMilestone.gsm in the General Examples/Events folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) contains an example of the use of Status elements.
specified manner. That is, the system includes a feedback loop that is used to control behavior.

**Read more:** Evaluating Feedback Loops (page 346).

A thermostat is the classic example of this (the heating and/or cooling rate is adjusted based on the current temperature). Another example might be a pond (the pumping rate to and from the pond is adjusted based on the water level).

The goal of a feedback control system is generally to control the value of a state variable in the system (typically represented by a Reservoir or a Pool in GoldSim) by actively adjusting additions and withdrawals to that variable so that it is maintained at or around a target value (e.g., the set temperature value on a thermostat).

**Read more:** Understanding State Variables in GoldSim (page 341).

In the case of the thermostat, the state variable is the amount of heat in the house (quantified via the temperature) and the additions are withdrawals are associated with turning on/off the heat or air conditioning. For a pond, the state variable is the amount of water in the pond (which could be quantified via the water level) and the additions are withdrawals are associated with turning on/off pumps that remove (or perhaps add) water.

If you are going to model such systems, you need to understand how to properly represent these controlling feedback loops in a realistic way. In particular, it is quite common for novice (and even some experienced) modelers to represent such systems in such a way that the model behaves quite poorly (e.g., oscillates unrealistically).

**Read more:** Avoiding Oscillatory Behavior When Using Reservoirs (page 248); Avoiding Oscillatory Behavior When Using Pools (page 269).

One of the best ways to avoid the potential for unrealistic oscillatory behavior (and to realistically represent how the system would be managed in the actual system) is use what is referred to as a “deadband” or a target range within which the system is controlled. This is actually how a thermostat typically works (e.g., “turn the heat on when the temperature is 68 degrees and turn it off when it reaches 70 degrees”). You could implement this in GoldSim using a Status element.

To illustrate this, let’s consider the example pictured below (you can find this example in Reservoir.gsm in the General Examples folder of your GoldSim directory, accessed by selecting File | Open Example... from the main menu):
In this simple model, water flows into a pond at a constant rate (100 m$^3$/day), and we wish to actively control the volume of water in the pond, maintaining the volume between a lower target or 290 m$^3$ and and upper target of 310 m$^3$ (the “deadband”). We do this by turning a pump (that removes water from the pond) on and off in a specified manner. The pump, when on, pumps at a rate of 200 m$^3$/day.

The Status element is used to determine if the pump is on or off. Its triggers are defined as follows:

**Note:** The triggers are defined as “On Stock” triggers, which force an unscheduled update at the point the condition is met, resulting in more accurate behavior. Using “On True” triggers would be less accurate (as unscheduled updates would not be created, and the trigger would not be pulled until the next scheduled update).

**Read more:** Specifying Triggering Events (page 356).

The Withdrawal from the pond is then defined as follows:
That is, when the pond volume is greater than the upper target volume, the pump is turned on. When it is below the lower target volume, it is turned off. Note that the Pumping Rate is larger than the Inflow, so when the pump is turned on, the volume in the pond is reduced. The essentially creates a feedback loop between the pond volume and the withdrawal – it is an active feedback control system that is analogous to those common in many engineered systems.

The pond volume behaves as follows, slowly moving between the two targets:

Milestone Elements

Milestone elements record the time that an event occurs. The Milestone element has three outputs: the Date (calendar time) that the triggering event occurred, the ETime (elapsed time) in the simulation that the triggering event occurred, and a Completion_Status (which is False prior to the element being triggered, and True afterward).

Read more: Setting the Basic Time Options (page 453).

You would use a Milestone element to record the time when a particular event occurred or condition was achieved. You could also use the Completion_Status of the Milestone to trigger other events (e.g., when Milestone X is reached, trigger event Y).

The dialog for a Milestone element looks like this:
The key inputs in the Milestone dialog are the triggering event(s) which cause the Milestone to be set. This is defined via a **Trigger**... button, which provides access to a standard Trigger dialog.

**Read more:** [Understanding Event Triggering](#) (page 355).

A Milestone also has several radio buttons and checkboxes.

If **Event may only occur once** is selected, then the Milestone records the time that it was first triggered. If it is triggered multiple times, a warning message is written to the Run Log.

If **Event may occur multiple times** is selected, then you must also select whether the Milestone is to **Store time of the first occurrence**, or **Store time of the last occurrence**. In the former case, the Milestone records the time that it was first triggered and ignores any other events. In the latter case, the Milestone records the time that it was last triggered.

If **Record a warning message if this milestone is never achieved** is checked, a warning message is written to the Run Log if the Milestone is never triggered.

To better understand how a Milestone element works, consider the following simple example. Assume that a Milestone element is triggered by an event that occurs every ten days, the Milestone is defined such that **Event may occur multiple times** is selected, and the Milestone is set to **Store time of the first occurrence**. The output of the Etime output of the Milestone in this case would look like this (note that in a chart, 1 corresponds to True, and 0 corresponds to False):
The Milestone records when it was first triggered (at 10 days), and sets its Completion Status to True (1).

Note that the initial value of the Etime output for a Milestone (i.e., its value prior to being triggered) is zero (0). Similarly, the initial value of the Date output for a Milestone (i.e., its value prior to being triggered) is the Start Date/time. This can be misleading, because if the Milestone is never achieved during a realization, the result will actually indicate that it was achieved at the beginning of the realization (Etime = 0). Such a result is particularly confusing when viewing probabilistic results.

To illustrate this, consider the following probabilistic result (accessed by selecting the ETime output and viewing the Distribution Result…):

This plot indicates that just under 20% of the time, the Milestone was achieved immediately. In reality, about 20% of the time, it was not achieved at all, but because the initial value for the time achieved was 0, the result indicates otherwise.
To get around this problem (and provide additional Milestone results), the Milestone element contains a **Results** tab:

At the bottom of the page are two buttons (**Dates...** and **Elapsed Time...**). Pressing these buttons displays the probability distribution of when the Milestone was achieved (in terms of calendar time or elapsed time, respectively) **considering only those realizations in which it was achieved**. Here is the **Elapsed Time...** result corresponding to the one shown previously:

As can be seen, the distribution no longer shows the (misleading) jump at ETime = 0; it only includes those realizations during which the Milestone was achieved.

**Note:** The **Dates...** and **Elapsed Time...** buttons are grayed out when the model is in Scenario Mode. That is, you cannot view these results when running and comparing scenarios.
Read more: Creating, Running and Comparing Scenarios (page 504).

The upper portion of the Milestone’s Results tab shows additional results of interest. Two of these results are defined with respect to a defined target time that you define (in terms of either Elapsed Time or Date). The target time defaults to the beginning of the simulation.

The results displayed directly below the target time are then defined as follows:

**Probability milestone achieved:** This is the probability that the Milestone is achieved during the simulation. Hence, this represents the fraction of realizations that are included in the Elapsed Time… or Date… plots accessed via the buttons at the bottom of the page.

**Probability milestone achieved by target time:** This is the probability that the Milestone is achieved by the specified target time.

**Mean lag (for late achievement):** For those realizations in which the Milestone is achieved after the target time (but before the end of the simulation), this is the mean time after the target time that the Milestone was achieved.

The browser view of a Milestone looks like this:

![Browser View of Milestone](image)

**Note:** Element inputs and outputs are only shown in the browser if you choose to Show Element Subitems (accessed via the browser context menu by right-clicking in the browser).

The example model StatusMilestone.gsm in the General Examples/Events folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) contains an example of the use of Milestone elements.

By default, Stochastic elements are sampled once per realization (when their parent Container is first activated, which is typically at the beginning of the realization).

Stochastics can also be triggered to resample themselves by events. This may be useful, for example, if you wanted to resample a parameter representing the value of a discrete change (e.g., the magnitude of a storm) every time a particular event occurred, or if you wished to generate stochastic time histories (e.g., a precipitation record) by sampling a Stochastic on a regular basis (e.g., every day).

The Resample… button in the Stochastic dialog provides access to a standard Trigger dialog for controlling when a Stochastic is sampled.

**Triggering a Stochastic**
Responding to Events

By default, Sampled once is selected and this button is grayed out. This implies that the element will be sampled only at the beginning of the simulation.

If you select the Resampled radio button, the Resample... button becomes available, and you can define one or more triggers.

**Note:** If the Stochastic is not contained within a conditional Container, or if it is contained in a conditional Container but is static (i.e., no triggers are defined and is not otherwise a function of time), it will automatically be sampled at the start of the realization (and never again). If it is contained in a conditional Container and is not static (i.e., has at least one trigger defined or is otherwise a function of time), it is automatically sampled when its parent Container activates, and is assigned a value of zero prior to activation of the Container.

**Read more:** Using Conditional Containers (page 899).

**Note:** One common requirement in models which represent a time-variable or stochastic process is to resample a Stochastic on a regular basis (e.g., every day or every month). You can do this in an easy way by utilizing the various Run Properties. For example, defining an On Changed trigger as “Day” would force the Stochastic to be triggered every day.

**Note:** If Stochastic X is correlated to Stochastic Y, you cannot specify that X is to be resampled (a Stochastic that is correlated can not be explicitly resampled). However, if Stochastic Y is resampled, then Stochastic X will also automatically be resampled (although this is not explicitly indicated in the dialog for X).

**Read more:** Understanding and Referencing Run Properties (page 486); Correlating Stochastic Elements (page 178).
The Interrupt element is triggered by an event, and responds by interrupting the simulation. The triggering event can be a discrete event signal or another type of event (e.g., a condition, such as X becoming greater than Y).

When the simulation is interrupted, you can display a user-defined message dialog with options for continuing or pausing (and subsequently aborting) the simulation and/or you can specify that the message is written to the run log. You also specify how to treat the interrupted realization if the simulation is resumed (e.g., continue realization, or keep current realization results and move to the next).

The Interrupt element has a number of potential uses:

- **To alert you to error conditions that indicate a logical problem in a model.** For example, if a certain variable should not go below 0 (if it does, it indicates a logic error in the model), you could use an Interrupt element to check for this condition. This could be used, for example, to test for material balance in a model that was simulating movement of material (e.g., water).

- **To help you debug a model that is crashing or showing unusual behavior.** For example, if a model crashes (or starts behaving unusually) when certain conditions are met or at a particular time in a simulation, you could use an Interrupt element to pause the simulation at that point so that you can browse the model and/or save the results up to that point in time.

- **To terminate (skip to the end) of a simulation when a certain set of conditions are met.** For example, if you were carrying out a Monte Carlo simulation of a project, the project may actually complete at a different time each realization. When the project completes, you could use an Interrupt element to skip to the end of the realization.

- **To display status information to a user.** In some cases, you may simply want to display status information to a user who is running GoldSim interactively (e.g., using the GoldSim Player). The message could, for example, prompt the user to pause the model, adjust some of the input parameters in a Dashboard, and then continue the simulation.

The dialog for an Interrupt element looks like this:
The sections below describe the use of the Interrupt element in detail.

The example file Interrupt.gsm in the General Examples/Events folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) includes a simple example on the use of Interrupt elements.

The key inputs to an Interrupt element are the interrupt triggers: the events and/or conditions that cause the simulation to be interrupted.

These are defined via the Trigger... button, which provides access to a standard Trigger dialog

Read more: Understanding Event Triggering (page 355).

The simulation will be interrupted whenever the trigger event occurs. Like all triggers in GoldSim, the Interrupt trigger can consist of multiple events and conditions.

In some cases, you may want to disable (turn off) an Interrupt element. When an Interrupt is disabled, it does not interrupt the simulation. You can disable the element by checking the box labeled Disable Interrupts by this Element at the bottom of the page.

When you do so, you will notice that the symbol for the element in the graphics pane (and browser) changes (to indicate that it is currently disabled):

![Interrupt Enabled](image1) ![Interrupt Disabled](image2)

Note: During a simulation, whenever a simulation is interrupted, you also have the opportunity to disable the Interrupt (i.e., ignore any Interrupts) for the remainder of the simulation.
Read more: Controlling the Message Dialog for an Interrupt (page 411).

The middle part of the Interrupt element is used to control the message dialog and the run log when a simulation is interrupted:

When defining an Interrupt, you can specify whether or not a message is displayed by selecting whether to Show Message. The default is Yes.

If you choose to show a message, you enter the message in the large input field directly below the Show Message field. The message field accepts formatted text (in Rich Text Format). Hence, you can cut and paste formatted text from another application into this box. You can also use the context menu for the input field to add formatting:

You can also specify whether or not additional information is included in the message dialog when the simulation is interrupted via the Add Element and Simulation Information to Message checkbox.

If this box is checked (the default), the message will look like this:
In addition to displaying the user-defined message, basic information about the simulation and element that caused the interruption is reported.

If the **Add Element and Simulation Information to Message** checkbox is cleared, the message will look like this:

![Message dialog](image)

**Note:** If you have added special formatting to the message, this will be ignored if the **Add Element and Simulation Information to Message** box is checked. That is, the formatting only appears in the message box if the extra simulation information is *not* written to the dialog.

Whenever a message dialog is displayed, you have the choice to **Ignore for remainder of simulation**. If you check this box and then continue (by either pressing the **Continue** button, or by pressing the **Pause** button and subsequently resuming the simulation using the Run Control toolbar), the Interrupt element that triggered the message will be ignored for the remainder of the simulation.

**Note:** If the message is also written to the run log, and is specified to be written as a Warning, the message dialog box displays a Warning symbol in the upper left-hand corner. Otherwise, it displays an Information symbol.

**Read more:** [Writing the Interrupt Message to the Run Log](page 414).

When a simulation is interrupted and a message dialog is displayed, two buttons are provided on the dialog: **Continue** and **Pause**:

![Message dialog](image)

Pressing the **Continue** button simply closes the dialog and continues the simulation. The simulation continues to the end (or until the next Interrupt occurs).

**Note:** If you check the **Ignore for remainder of simulation** box and then press the **Continue** button, the Interrupt element that triggered the message will be ignored (disabled) for the remainder of the simulation.

Pressing the **Pause** button closes the dialog and pauses the simulation. When a simulation is paused, this will be indicated in the status bar at the bottom of the GoldSim window:
The Run Control toolbar can then be used to control the simulation:

![Run Control Toolbar]

The options (from left to right) consist of aborting the simulation, resuming the simulation (which is equivalent to pressing the **Continue** button in the Interrupt message dialog), stepping one timestep at a time, or stepping one realization at a time.

*Read more:* [Pausing and Stepping through a Simulation](#) (page 502).

Once a model is paused, you can browse it and explore results in largely the same manner you would if you were in Result Mode.

While a model is paused, you may decide you want to ignore future interrupts before resuming the simulation. You can do this by navigating to the Interrupt element, right-clicking on it, and selecting **Ignore Interrupts** from the context menu before resuming the simulation.

You can control what part of the model is displayed in the graphics pane when you press the **Pause** button in the message dialog using the **On Pause, Select this Element** field in the Interrupt properties dialog:

![Interrupt Properties Dialog]

The drop list always contains two options: "None" (the default), and "This Interrupt Element". If "None" is selected, when you press the Pause button in a message dialog, the model view does not change from the view when the simulation was started. If "This Interrupt Element" is selected, when you press the Pause button in a message dialog, GoldSim displays the Container in which the Interrupt element resides. The drop list also contains any elements that are directly referenced in the trigger dialog for the element. If one of these elements is selected, when you press the Pause button in a message dialog, GoldSim displays the Container in which that element resides.

If you choose to Abort the simulation, GoldSim will give you a choice of keeping results or discarding results (and returning to Edit Mode). If you are running multiple realizations and have paused after the first realization, a checkbox will be provided to give you the option of excluding the current (partially complete) realization, or including it. If you choose to include it, GoldSim will skip to the end of the realization (i.e., skip all further calculations for the realization, keeping the values of all elements constant at their last computed value).

**Note:** If you have aborted in the *first* realization and choose to save results, GoldSim will automatically skip to the end of the realization (i.e., skip all further calculations for the realization, keeping the values of all elements constant at their last computed value).

*Read more:* [Aborting a Simulation](#) (page 503).
Warning: When a simulation is interrupted, in almost all cases it will be interrupted in the middle of an update. That is, GoldSim elements are, by necessity, computed in a specific order (referred to as the causality sequence). When an Interrupt is triggered, it will be done in the middle of this sequence. That means that some of the elements will have been updated for the current timestep, and others will not have been updated yet. Hence, it may be necessary to view the causality sequence when browsing a model to fully understand the current values that are displayed.

Read more: The Causality Sequence and Element Updating (page 343).

Warning: If you abort a simulation and include the results from the last (partially complete) realization, you should be careful with how you interpret the results. In particular, because the last realization was not complete, it may be inappropriate to combine it with or compare it to other (full) realizations.

In some cases, in addition to (or instead of) displaying a message when an interrupt occurs, you may want to also write a message to the run log.

Read more: The Run Log (page 548).

This is controlled by the Add to Run Log field in the Interrupt property dialog:

The drop list contains three options:

- "No". This is the default. The message is not written to the run log.
- "As Message". The message is written to the run log.
- "As Warning". The message is written to the run log as a Warning. As a result, a dialog is displayed at the end of the simulation indicating that the simulation triggered a warning.

Note: If you choose to both display a message and write to the run log, the Add to Run Log field has a small impact on the message dialog displayed when a simulation is interrupted. In particular, if the message is written to the run log as a warning, the message dialog box displays a Warning symbol in the upper left-hand corner of the message dialog. Otherwise, it displays an Information symbol.

In some cases, you may choose not to show a message when an Interrupt is triggered. As discussed below, this is useful, for example, if you are using the Interrupt to terminate a realization and skip to the end when a certain condition occurs (e.g., a project completes). In other cases, it may not be possible to
display the message (e.g., when simulating scenarios, or when carrying out a
distributed processing simulation, a sensitivity analysis, or an optimization).

You must specify how you want GoldSim to handle the Interrupt when the
message is turned off or cannot be displayed. This is done via the **Specify
Action When Message is off or cannot be shown** field at the bottom of the
Interrupt property dialog.

The drop list contains five options:

- "Continue Simulation". This is the default. The simulation continues.
- "Skip remainder of current Realization and Continue". GoldSim skips
to the end of the realization (i.e., it skips all further calculations for the
realization, keeping the values of all elements constant at their last
compute value), and continues on to the next realization.

**Note:** This option is what you should select if you are using the Interrupt to
terminate a realization and skip to the end when a certain condition occurs (e.g.,
a project completes).

- "Skip remainder of current Realization and Abort". GoldSim skips to
the end of the realization (i.e., it skips all further calculations for the
realization, keeping the values of all elements constant at their last
compute value), and then aborts (keeping results for all realizations up
to that point, including the partially completed realization).
- "Discard current Realization and Abort". GoldSim discards the current
(partially completed) realization, and then aborts (keeping results for all
realizations up to that point, excluding the partially completed
realization).
- "Abort and Return to Edit Mode". The simulation is aborted and the
model is returned to Edit Mode.

**Warning:** When running distributed processing, scenarios, sensitivity analysis
or an optimization, Interrupt messages are never displayed. Moreover, unless
the **Specify Action When Message is off or cannot be shown** field is set to
"Continue" or "Skip remainder of current Realization and Continue", the
Interrupt will result in a fatal error.

**Note:** Even if the message is turned off (or cannot be displayed), you can still
specify that the message is to be added to the run log. Note, however, that a run
log is not produced for sensitivity analyses and optimizations.

**Read more:** [*Writing the Interrupt Message to the Run Log*](page 414).

### Generating Discrete Changes Using Time Series Elements

Time Series elements can be used to generate a series of discrete changes.

**Read more:** [*Using Time Series Elements*](page 186).
In order to do so, you must specify that the data that you are entering represent a series of discrete changes. To change the data type, press the Type... button from the Time Series dialog, and the following dialog will be displayed:

Select “Discrete Change” as the Type. After doing so and closing the dialog, the main dialog for the Time Series will indicate that the Time Series now Represents a “Discrete Change (Instantaneous)”: 

Under these circumstances, the output produced by the Time Series element is actually a discrete change signal. You specify the value for each signal, as well as the time that the signal is to be emitted.

For example, assume that the data series for a Time Series was defined as follows:

In this case, four discrete change signals would be emitted by the element. Discrete change signals emitted by a Time Series element always carry an “Add” instruction.

Time Series elements that generate discrete change signals have the ability to influence when a model is updated. Typically, a model is updated at every “scheduled” timestep. That is, all the elements are computed at every timestep. However, a Time Series element can force a model to be updated between “scheduled” timesteps. For example, suppose that you have specified a 10 day timestep. If a Time Series element has a discrete change that was specified to occur at, say 23 days, GoldSim would insert an update (an unscheduled update) between timesteps (i.e., at 23 days) in order to more accurately represent the event.

Read more: Understanding Timestepping in GoldSim (page 455).
Warning: Time Series elements are designed to interrupt the clock and insert a new update when the discrete change is released. If you choose to disable unscheduled updates, a discrete change that falls between scheduled updates will be deferred to the next scheduled update. If multiple discrete changes occur between scheduled updates, they will be summed and released at the next scheduled update.

Read more: Controlling Unscheduled Updates (page 470).

The example model BasicTimeSeries.gsm in the General Examples/TimeSeries folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) contains an example of the use of Time Series elements to generate discrete change signals.

How GoldSim Inserts Events into a Simulation

When a discrete event is generated in GoldSim by certain types of elements (e.g., a Timed Event, an Event Delay, or a Discrete Change Delay), the events may not fall exactly on a “scheduled” timestep (i.e., a timestep that was defined in the Time tab of the Simulation Settings dialog). That is, the events may actually occur between scheduled timesteps.

These events trigger an “unscheduled update” of the model. Unscheduled updates are timesteps that are dynamically inserted by GoldSim during the simulation in order to more accurately simulate the system. That is, they are not specified directly prior to running the model, GoldSim inserts them automatically (and, generally, without you needing to be aware of it).

For example, if you had specified a one day timestep, and a Timed Event occurs at 33.65 days (i.e., between the scheduled one-day updates), GoldSim would insert an unscheduled update at 33.65 days.

Read more: Understanding Timestepping in GoldSim (page 455).

By default, scheduled updates are always dynamically inserted by GoldSim. However, in some (rare) cases, you may want to prevent unscheduled updates from being inserted. For example, if your model included a specialized algorithm that was designed based on the assumption that the timestep was constant, inserting unscheduled updates could invalidate the algorithm. To support such situations, GoldSim allows you to disable unscheduled updates.

Warning: Because unscheduled updates are intended to more accurately represent a complex dynamic system, disabling this feature should be done with caution, and is generally not recommended.

Read more: Controlling Unscheduled Updates (page 470).

Determining if an Event Has Occurred

GoldSim provides a special function that can be useful in discrete event modeling: the "Occurs" function. The Occurs function accepts a single argument which must be a discrete event signal.
The Occurs function returns a condition (True or False). If the argument (the discrete event signal) has occurred during the current update, the value is True, otherwise it is False.

This can be useful in situations where you need to reference whether or not two events have occurred simultaneously. For example, suppose that there are two events in your simulation (Event1 and Event2), and you want to trigger a response if and only if Event1 occurs and Event2 does not (i.e., you do not want to trigger the response if both events occur simultaneously). A good way to do this is to use the Occurs function in the Required Condition field for each element.

The Trigger dialog that captures this behavior would look like this:

![Image of the Trigger dialog]

**Note:** If you try to plot an Expression defined using the occurs operator, the Expression will always show False (unless the event occurs exactly at a scheduled update point), since by default, GoldSim only records results at the scheduled plot points, and not at the unscheduled timesteps inserted by events (which are the only points at which the Expression would be evaluate to True). Note, however, that GoldSim provides an option to include unscheduled updates in time history displays (under a specified set of conditions) in the Advanced Time settings.

**Read more:** [Understanding Timestepping in GoldSim](#) (page 455); [Including Unscheduled Updates in Time History Results](#) (page 477).

### Controlling the Calculation Sequence of Events

When you build a model, GoldSim automatically *sequences* the elements in the order that they must be computed. For example if A was a function of B, and B was a function of C, C would be sequenced first, followed by B, followed by A. This is referred to as the causality sequence. In this simple example, the sequence is obvious, but for complex models with looping logic the causality sequence may not be readily apparent.

In most cases, you do not need to be concerned with how GoldSim sequences the elements. However, in some cases (particularly when simulating systems
that include discrete changes and looping logic), expert users may need to understand or manipulate the causality sequence. (The causality sequence can be viewed by selecting **Model|Causality Sequence** from the main menu.)

**Read more:**  The Causality Sequence and Element Updating (page 343).

Problems arise when simulating looping systems and/or systems with discrete changes because in these cases the appropriate order of the calculations can be ambiguous. A simple example is shown below:

For this model, in what order should the elements be calculated?

For these situations, GoldSim provides some special tools to force the sequencing to occur in a particular order.

**Read more:**  Addressing Ambiguous Causality Sequences (page 971).
Chapter 6: Customizing the Interface

As You Like It.

William Shakespeare

Chapter Overview

GoldSim allows you to customize the user interface in a variety of ways in order to best match the way that you use the program. For example, you can modify and create custom toolbars, change the appearance of the graphics pane, and modify the appearance of elements and links. You can also create your own custom units. This chapter describes how you can control and customize these user interface components.

In this Chapter

The following topics are discussed in this chapter:

- Customizing Toolbars
- Customizing the Appearance of the Graphics Pane
- Editing the Appearance of Elements
- Viewing and Creating Units
Customizing Toolbars

GoldSim has a number of toolbars that you can use to carry out common functions. The available toolbars are as follows:

- **Standard toolbar**: provides buttons for common actions (e.g., save a file, search, copy, paste, access simulation settings, access Help).

- **Run Control toolbar**: provides buttons for running your model.

- **Element toolbar**: provides buttons for adding elements to your model.

- **Advanced toolbar**: provides buttons for carrying out advanced options (e.g., distributed processing, running scenarios, running sensitivity analyses).

- **Drawing Tools toolbar**: provides buttons for adding graphical objects to your models.

- **Graphical Tools toolbar**: provides buttons for grouping, aligning, rotating, spacing, resizing and layering objects.

- **Dashboard Tools toolbar**: provides buttons for adding Dashboard controls.

You can turn these toolbars off and on, edit existing toolbars, create new (custom) toolbars, and move them around the screen.

When you open GoldSim for the first time, all the toolbars are visible. Most of the toolbars are originally docked at the top of the screen, and two of the toolbars (Drawing Tools and Dashboard Tools) are originally docked (vertically) to the right of the graphics pane.

All toolbars can be hidden (deactivated). To activate or deactivate a toolbar, right-click anywhere within a toolbar (or the main menu). The following context menu is displayed:
Activated toolbars have a check mark next to them. To activate or deactivate one of the toolbars, click on it.

The toolbars (and any menu bars) are originally docked. You can undock them and move them to another location on the screen (e.g., the bottom, or to the right of the graphics pane) by grabbing the docking grab bar at the left (or top) of the toolbar/menu bar and dragging it to the desired location:

If you drag it to the edge of the window it will dock itself at that edge (otherwise, it will float). Note, however, that you cannot dock a toolbar on the left side if the window. That is, you can only dock at the top, the the right, and at the bottom (above the status bar).

You can also undock a toolbar or menu bar (and turn it into a floating window) by double-clicking on the docking grab bar.

Customizing the Appearance of the Graphics Pane

GoldSim provides a number of options for customizing the graphics pane:

- You can add a background color to each Container;
- You can add a grid to each Container;
- You can adjust the size of the graphics pane in each Container;
- You can edit the appearance (e.g., shape, color) of influences in the Container; and
- You can filter (hide) influences of certain types.

In addition, you can copy the graphical settings (e.g., background color, influence shape) for one Container to other Containers in your model.

These options are described in detail in the sections below.

The Container dialog can be accessed by right-clicking anywhere in the graphics pane and selecting Properties... from the context menu, by right-clicking on a Container and selecting Properties, or by double-clicking on a Container.

The Graphics tab of the Container dialog provides access to options for controlling the appearance of the graphics pane for the Container, including viewing a grid and adding a background color:
Note that right-clicking anywhere in the graphics pane and selecting **Properties** from the context menu at the highest level of the model (the main Container) displays the properties for this highest level Container (referred to as the Model Container). When you do this, you will also see additional tabs for the various simulation settings for the model:

Checking the **Show Grid** checkbox for a Container (which is cleared by default) adds a grid to the graphics pane:
The grid spacing is fixed (it is a function of the size of the graphics pane) and cannot be modified.

Checking **Snap to Grid when dragging components** (which is checked by default) causes the center of objects (elements and graphic objects) to snap to the grid when they are moved in the graphics pane.

The **Color** drop-list provides a list of colors that can be applied to the background for the Container’s graphics pane.

---

**Note:** When you insert a new Container into a model, it inherits the properties (e.g., grid setting and background color) of its parent Container. However, this inheritance is only applied when the new Container is created, and the properties of the new Container are not linked to the properties of the parent (e.g., if you subsequently change the parent’s background color, it will not change the background color of any existing child).

---

### Adjusting the Size of the Graphics Pane

If you scroll far enough horizontally or vertically within the graphics pane you will eventually reach the “edge” of the graphics pane. That is, the graphics pane represents a document of fixed size.

By default the graphics pane has a width of 17 inches and a length of 22 inches (i.e., for any Container, if you view the graphics pane at a 100% scale, it will appear to be 17 inches wide and 22 inches long, although you will have to scroll to view the entire document).

If you would like to change the size of the graphics pane, you can do so from the **Graphics** tab of the Container dialog.
Customizing the Appearance of the Graphics Pane

**Note:** The Container dialog can be accessed by right-clicking anywhere in the graphics pane and selecting **Properties**… from the context menu, by right-clicking on a Container and selecting **Properties**, or by double-clicking on a Container.

At the top of this tab are several options related to adjusting the size of the graphics pane:

The **Width** and **Height** fields can be used to change the size of the graphics pane. These dimensions can be specified in inches, centimeters or millimeters (as defined in the **Units** field). If **Show Printer Page Boundaries** is checked, the boundaries of the printed pages in the graphics pane (as determined by your printer settings) will also be shown in the graphics pane.

**Note:** When you insert a new Container into a model, it inherits the properties (e.g., size of graphics pane) of its parent Container. However, this inheritance is only applied when the new Container is created, and the properties of the new Container are not linked to the properties of the parent (e.g., if you subsequently change the graphics pane size in the parent, it will not change the graphics pane size in any existing child).

**Saving the Graphics Pane's Position and Scale**

Whenever you leave one Container in the graphics pane and enter (i.e., view the contents of) another, by default GoldSim remembers the position and scale for the previous Container, and this information is written to the file when you save it. Hence, when you (or someone else to whom you have given the file) enter a Container, it is automatically displayed at the same position and scale that it was last viewed.

**Note:** If you clear the **Use saved container scale and position** checkbox in the **Graphic** tab of the Options dialog (accessed via **Model|Options**…), this information will be ignored, and all Containers will be displayed at 100% scale in the upper left-hand corner of the page. This setting is saved to the Windows Registry and is therefore applied to all GoldSim files that you open on your machine.

**Editing the Appearance of Influences**

GoldSim allows you change a number of the graphical properties of an influence. You can change its shape, thickness and color, and add text to the influence. In addition to editing each influence separately, you can also modify the appearance of all influences in a Container from the Container dialog.

The details of how you can edit the appearance of influences is described in the sections below.

By default, all links are initially drawn as curves (but are drawn with no curvature). You can add curvature to the link by selecting it and dragging the control point with your cursor.
You can further modify the appearance of influences by changing their color and thickness, and changing their shape (straight lines or orthogonal lines). You can do so by right-clicking on an influences, which will cause the following menu to be displayed:

- **Shape**
- **Color**
- **Width**
- **Order**
- **Add Label**

The Shape selection provides three options: **Straight**, **Orthogonal** and **Curved**. Orthogonal influences consist only of vertical and horizontal lines:

The Color selection allows you to choose from a variety of colors. The Width selection allows you to choose from 1 point (the default) to 6 points.

**Note:** You can globally modify the appearance of all influences in a Container (or your entire model) based on the type of link they represent (e.g., values or conditions). This includes changing the default for all new influences that are drawn.

**Creating a Segmented Influence by Adding Nodes**

The shape of an influence can be set to either **Straight**, **Orthogonal** or **Curved**.

**Read more:** [Controlling the Appearance of All Influences in a Container](page 428).

**Read more:** [Changing the Shape, Thickness and Color of an Influence](page 426).

If the shape of an influence is defined as **Straight**, GoldSim allows you to add nodes to the influence, such that it is turned into a segmented line:
You add a node to an influence by clicking on it while holding the Ctrl key. You can add as many nodes as desired. When you select the influence, the nodes will be shown:

To delete a node, click on it while holding the Ctrl key.

Once a node has been added, you can move a node by dragging. You can also move the entire influence by grabbing within a segment and dragging.

Note: If you change the shape of an influence to Orthogonal or Curve, all nodes are immediately deleted.

Note: If Snap to Grid when dragging components is selected for the graphics pane, the segment or node will snap to the grid as you drag it.

Adding Text to an Influence

You can add text to an influence by right-clicking on the influence and selecting Add Label from the context menu, or Shift+double-clicking on the influence (double-clicking on the influence while pressing the Shift key). This will add a textbox to the influence with the word "Text" inside (which you can subsequently replace with your own text).

You can edit the properties of the label (e.g., font, color, outline) by right-clicking on the influence and selecting Properties…. This dialog is similar to that used to edit the properties of text objects in the graphics pane.

Read more: Changing the Appearance of Text Objects (page 748).

To delete the text associated with an influence, select the text box, and press the Delete key.

The Graphics tab of the Container dialog provides access to options for controlling the default appearance of all influences in the Container:
Customizing the Appearance of the Graphics Pane

Note: The Container dialog can be accessed by right-clicking anywhere in the graphics pane and selecting Properties… from the context menu, by right-clicking on a Container and selecting Properties, or by double-clicking on a Container.

The section of the dialog labeled “Influences” allows you to adjust the Influence Shape, Color and Width of all influences in the Container. The appearance of the influences can be specified differently depending on the type of output(s) in the link(s) represented by the influence. By default, there are four types of outputs that can be represented in a link: Values (the most common type of output); Conditions (i.e., True/False); Transactions (discrete event and discrete change signals); and Multiple Types (for an influence that contains more than one type of output).

Note: Some GoldSim extension modules (e.g., the Contaminant Transport Module) may add additional output types to this list.

Warning: Although you can change the influence colors for the various types of outputs here, in most cases, you should refrain from doing so. This is because others looking at your model will be used to the visual cues provided by the default colors (e.g., dark gray for values, green for conditions, red for transactions), and changing these could make your model more difficult for others to understand.

If you change the Influence Shape, or the Color or the Width for an Output Type, and then press OK, GoldSim will modify all of the influences in the Container, with the following exceptions:
• GoldSim will not modify the appearance of influences inside any Containers within the Container being edited.

• If you have previously changed the appearance of an influence in the Container manually (by right-clicking on it) such that it is different from the default settings in the Container dialog, GoldSim will treat these as “customized influences” and will not reset them to the new defaults unless you first check the Reset customized influences checkbox in the dialog before pressing OK.

Note: When you insert a new Container into a model, it inherits the properties (e.g., Influence Shape, Colors and Widths) defined for its parent Container. However, this inheritance is only applied when the new Container is created, and the properties of the new Container are not linked to the properties of the parent (e.g., if you subsequently change the parent’s Influence Shape, it will not change the Influence Shape of any existing child Container).

In addition to controlling the appearance of all the influences in a Container, GoldSim also provides the ability to copy the influence appearance settings from one Container to other Containers in your model.

Read more: Copying Container Settings to Other Containers in a Model (page 432).

Finally, GoldSim provides you with the ability to temporarily filter (hide) influences in a Container.

Read more: Filtering Influences (page 430).

Filtering Influences

In some situations, you may wish to temporarily "hide" or "filter" some types of influences in order to simplify the graphics pane. When you filter an influence, the link(s) within the influence still exist, but the influence is not shown on the graphics pane.

Influence filters are defined separately for each Container from the Graphics tab of the Container dialog:
Customizing the Appearance of the Graphics Pane

The Container dialog can be accessed by right-clicking anywhere in the graphics pane and selecting **Properties**... from the context menu, by right-clicking on a Container and selecting **Properties**, or by double-clicking on a Container.

The section of the dialog labeled “Influences” allows you to **Filter (Hide)** all of the influences in the Container. The filter can be specified differently depending on the type of output(s) in the link(s) represented by the influence. By default, there are four types of outputs that can be represented in a link: Values (the most common type of output); Conditions (i.e., True/False); Transactions (discrete event and discrete change signals); and Multiple Types (for an influence that contains more than one type of output).

**Note:** Some GoldSim extension modules (e.g., the Contaminant Transport Module) may add additional output types to this list.

For example, if the **Filter (Hide)** checkbox was checked for “Conditions”, any influences in the Container that just represented Condition outputs would be hidden.

When you press **OK** to close the dialog, the filter is applied. You can toggle the filter off (or on) by pressing **Ctrl+H**, by selecting **View|Filter** from the main menu or pressing the Filter button in the Standard toolbar. When the filter is on, the button will appear depressed.
Customizing the Appearance of the Graphics Pane

**Note:** When you insert a new Container into a model, it inherits the properties (including the filter settings) defined for its parent Container. However, this inheritance is only applied when the new Container is created, and the properties of the new Container are not linked to the properties of the parent (e.g., if you subsequently change the parent’s filter settings, it will not change the filter settings of any existing child Container).

In addition to defining the manner in which influences are filtered in a Container, GoldSim also provides the ability to copy the influence filter settings from one Container to other Containers in your model.

**Read more:** [Copying Container Settings to Other Containers in a Model](#) (page 432).

When you insert a new Container into a model, it inherits the properties (e.g., background color, influence appearance) of its parent Container. However, this inheritance is only applied when the new Container is created, and the properties of the new Container are not linked to the properties of the parent (e.g., if you subsequently change the parent’s background color, it will not change the background color of any existing child).

In some cases, however, you may wish to apply (copy) one or more of the graphical properties that you have defined for one Container to another existing Container in your model. GoldSim provides the ability to do this.

At the bottom of the **Graphics** tab of the Container dialog, GoldSim provides a text button allowing you to **Click to apply these settings to other containers or subcontainers**:

![Click to apply these settings to other containers or subcontainers](image)

**Note:** The Container dialog can be accessed by right-clicking anywhere in the graphics pane and selecting **Properties**... from the context menu, by right-clicking on a Container and selecting **Properties**, or by double-clicking on a Container.

When you click on this text, the following dialog is displayed:
In the top part of the dialog, you specify the Container to which you wish to copy the graphical settings. If you click the **Apply to nested containers** checkbox, the settings will also be copied to all of the nested subcontainers within the selected Container.

**Note:** If you wish to copy the settings to all the Containers in your model, select the main Container (called “Model”), and check the **Apply to nested containers** button.

The “Update Attributes” section of the dialog allows you to select which settings you would like to apply. This is a tree that expands. If you want to apply all of the attributes at a particular level, check the box at the branch (e.g., Update Graphic Pane Attributes) and it will select all the items below it. Alternatively, you could open a branch, and clear some of the checkboxes so only certain attributes are copied:

After selecting the Container(s) to copy the attributes to, and the attributes to be copied, click the **OK** button to copy the settings.
Editing the Appearance of Elements

You can edit the appearance of an element by clicking on the Appearance… button in the element's properties dialog, or selecting Appearance… from the context menu for the element.

This provides access to the Appearance dialog:

The dialog is tabbed. The various tabs are discussed below.

Changing the Element’s Symbol

If you select the User-defined symbol radio button in this tab of the dialog, you will be prompted for the name of a file to be used for the symbol.

The file must be an enhanced Windows metafile (.emf). Select a file and press OK. If the new image is a different size than the old image (which will typically be the case), you will be asked whether you want to adjust the size of the new image to match the size of the existing element or to adjust the size of the element to match the size of the new image.

GoldSim utilizes enhanced metafiles (.emf) for element symbols because they are vector graphics (and therefore scale without losing image quality). Most advanced graphics programs can create enhanced metafiles (or convert other formats to an enhanced metafile). Note, however, that unless you create the original image using a vector format (i.e., an enhanced metafile), it will not be a true vector graphic and will lose image quality when scaling.
You can use GoldSim's graphic capabilities to convert other graphics formats to an enhanced metafile as follows:

1. Insert a graphic image (e.g., a bitmap) into the graphics pane;
2. Select the image and click **Graphics | Export…** from the main menu (or press **Ctrl+E**);
3. Specify that you wish to save the file as an enhanced metafile.

*Read more: [Adding Images](page 753).*

Once the image for the element is defined using a user-defined symbol, you can change the symbol by pressing **Select new Image…**, or you can switch back to the default image provided by GoldSim by selecting the **Default symbol radio** button.

You can also change or reset the images for multiple elements in a model (e.g., all elements of one type) by selecting **Graphics|Change Symbols…** from the main menu.

Two additional buttons are also present on the **Image** tab:

- **Reset Size:** This resets the size of a scaled image back to its original size.
- **Reset Ratio:** This resets the aspect ratio of a scaled image back to its original ratio (by changing the symbol’s height).

In some situations, you may wish to reset the images for all elements in a model (or in a particular Container) to a custom image or (if they have already been customized) back to the default symbol. You can accomplish this by selecting **Graphics|Change Symbols…** from the main menu, which will display a dialog for globally changing and resizing symbols.

Within this dialog, you first select whether you wish to **Change Symbol** and/or **Change Symbol Size** for the selected elements. When changing a symbol, you can either revert the selected elements to the original (default) symbol (assuming you have previously modified the symbols in some manner), or you can select a user-defined symbol.

If you wish to change the symbol to a user-defined symbol, the file must be an enhanced Windows metafile (.emf). GoldSim utilizes enhanced metafiles for...
element symbols because they are vector graphics (and therefore scale without losing image quality). Most advanced graphics programs can create enhanced metafiles (or convert other formats to an enhanced metafile). Note, however, that unless you create the original image as an enhanced metafile, it will not be a true vector graphic and will lose image quality when scaling.

When changing the symbol size, you can choose to either scale the image to a **Percent of Default** (a fraction of the element’s default image size) or directly specify the **Pixel Width** (the width of the image in pixels).

After specifying the type of change you wish to make, you must specify the scope of the changes (i.e., what elements the changes will be applied to). The **Scope of changes** drop-down list has up to four options:

- **Selected elements only**: Only the elements that have been selected (prior to opening the dialog) are modified. This option is only available if one or more elements have actually been selected.
- **Current container only**: Only the elements in the current Container (but not any elements in any child Containers) are modified.
- **Current container only + children**: Only the elements in the current Container (and any elements in child Containers) are modified. This option is only available if you are inside a Container that contains other Containers.
- **Entire model**: All elements in the model are modified.

The scope of the changes can be further refined by specifying the specific element types to which the modification will be applied. This is done by checking or clearing the boxes next to each element type in the **Element types affected** portion of the dialog.

Two tabs within the Appearance dialog of an element control the appearance of the element's label. The **Font** tab controls the label's font.

The options in this dialog are self-explanatory.

The **Labels** tab controls the position of the label relative to the symbol.
You can choose one of nine positions for the label. Note that if one of these nine positions is not sufficient, you can also manually move the label by selecting an **X-Offset** and/or a **Y-Offset** (these can be negative numbers). In addition, if **Allow Label Move** is checked in the **Locking** tab, you can select and manually move the label.

**Read more:** Limiting the Changes That Can Be Made to the Element’s Appearance (page 438).

Two tabs within the Appearance dialog of an element control the appearance of the element's background and outline. The **Line** tab allows you to draw a line (a box) around the symbol and label.

By default, the line is transparent. You can, however, turn off the transparency and then edit the line's color, style (e.g., solid, dashed, etc.) and width (number of points or pixels):

The **Fill** tab allows you to change the appearance of the fill for the symbol and label.
The fill consists of a background and a foreground. The **Foreground Color** is superimposed on top of a **Background Color** according to a selected **Pattern**. By default, both are transparent. By changing these, you can add a fill:

![Image of TimeSeries1 element]

**Note**: The most common reason to add an outline or a fill to an element is to highlight it for a particular reason (e.g., to remind yourself that it needs to be modified, or perhaps to highlight key assumptions).

### Limiting the Changes That Can Be Made to the Element’s Appearance

The **Locking** tab is used to control the types of changes that can be made to the appearance of the element:

![Appearance dialog with Locking tab]

The options are as follows:

- **Lock Appearance**: If you check this box and then close the dialog, you can no longer move or scale the element in any way (i.e., all of the subsequent options in this dialog are grayed out).

- **Allow Move**: If this box is checked (the default), you can move the element around within the graphics pane. Otherwise, its position remains fixed.
Allow Label Move: If this box is checked, you can select and move the element's label (separately from the element). If the box is cleared (the default), the label can only be moved with the element symbol itself.

Allow Scaling: If this box is checked (the default), you can change the size of the object by selecting it and dragging one of the control handles (small boxes on the edges of the image).

Lock Aspect Ratio: If this box is checked (the default), the aspect ratio of the image is locked when it is scaled. Otherwise, you can stretch the image and change the aspect ratio.

Viewing and Creating Units

One of the more powerful features of GoldSim is that it is dimensionally-aware. You enable this capability by assigning display units (and hence dimensions) to the elements (and hence to the inputs and outputs) of your model. You can enter data and display results in any units

GoldSim has an extensive internal database of units (all of these units are listed in Appendix D). You can also create your own units.

When you create a link, GoldSim ensures dimensional consistency and carries out all unit conversions internally. For example, you could add feet and meters in an expression, and GoldSim would internally carry out the conversion. (If you tried to add feet and seconds, however, GoldSim would issue a warning message and prevent you from doing so.)

Read more: Using Dimensions and Units (page 89).

The GoldSim Units Manager allows you to view all of the units that GoldSim recognizes. You can also use the Units Manager to add new units if necessary.

You can access the Units Manager by selecting Model|Units... from the main menu, which will display the following dialog:

The GoldSim Units Manager

Clicking the Plus sign next to a category expands the category to show its units.
There is also a button in the Advanced toolbar for accessing the Units Manager:

The Units Manager dialog initially displays the *unit categories* in the model. A unit category has a name and a specific set of dimensions (displayed in the Description column). The abbreviations for the dimensions are as follows:

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mass</td>
<td>M</td>
</tr>
<tr>
<td>Length</td>
<td>L</td>
</tr>
<tr>
<td>Time</td>
<td>T</td>
</tr>
<tr>
<td>Temperature</td>
<td>t</td>
</tr>
<tr>
<td>Current</td>
<td>C</td>
</tr>
<tr>
<td>Amount*</td>
<td>A</td>
</tr>
<tr>
<td>Luminosity</td>
<td>l</td>
</tr>
<tr>
<td>Angle</td>
<td>&lt;&gt;</td>
</tr>
<tr>
<td>Currency</td>
<td>$</td>
</tr>
</tbody>
</table>

*Used for a quantity of matter (i.e., moles)

**Note:** A category does not need to have dimensions. It can be dimensionless.

If you expand a category (by clicking on the category folder's "+") , the units in that category are displayed:

When a category is expanded to show units, the first column contains the unit's abbreviation, the second contains its description, and the third contains the
conversion factor between the unit and the "internal units" for that category (e.g., SI units).

By default, GoldSim only shows the primary SI unit (e.g., m for length). However, GoldSim recognizes all of the SI unit prefixes. If you check the **Show all SI unit prefixes** box, they will also be displayed:

![Image of GoldSim Units Manager dialog]

*SI units are displayed in blue in the dialog.*

---

**Note:** The **Remove** button on the dialog is for deleting units. However, you cannot delete the built-in units (and the button is greyed out when these units are selected). Only custom units that you have created can be removed.

---

**Creating New Units**

Occasionally, the internal units supplied by GoldSim may not be sufficient, and you will want to create your own custom unit. You can do this by pressing the **Add** button in the Units Manager dialog (accessed via **Model|Units…** from the main menu). This will display the following wizard for creating a new unit:

![Image of Create Custom Unit wizard]

This wizard will walk you through the steps required to add or edit a custom unit in GoldSim. To begin, enter a short abbreviation for the unit (8 character maximum), as well as a short description.
To create a new unit, enter the alphanumeric Abbreviation for your new unit, along with a brief Description string. Pressing Next> will bring up the second page of the wizard:

In this dialog, you must enter an expression that defines the new unit in terms of existing units. For example, if you were defining a new unit for length that was equal to 5 feet and 7 inches, you would type in 5 ft + 7 in.

After defining the unit, pressing Next> will bring up the third (and final) page of the wizard:

All units are placed into Categories, which are used to group the units within the Units Manager (and in context menus for units in element dialogs). When you define a new unit, GoldSim will display existing categories that have the appropriate dimension.

**Note:** The dimensions for a Category do not have to be unique. For example, the Categories Displacement, Length and Wavelength could all have dimensions of L (length).

If you select “New Category” (the only option if there are no existing Categories with the appropriate dimensions), the dialog prompts you for the name of the Category:
After selecting an existing Category or defining a new Category, press **Finish**. When you do so, the new unit (and the unit Category) will be marked in red to indicate that a custom unit has been defined:

You may want to add a new unit if you wish to use a different abbreviation in your expressions. For example, if you wished to use the unit “year” to denote a year of time, you could create a new unit with an abbreviation of ”year” (the built-in abbreviations for a year of time in GoldSim are yr and a).

**Note:** Due to the special nature of currency units (unique, constant conversion rates cannot be assigned), GoldSim provides a special mechanism for creating and editing currency units. In particular, a special dialog for specifying and editing currency units can be accessed by pressing **Model | Currencies…** from the main menu. You cannot edit conversion rates in the Units Manager. Specifying and editing currency units is described in detail in Chapter 3 of the **GoldSim Financial Module User's Guide**.
The **Remove** button allows you to delete user-defined units (you can only delete user-defined units; built-in units cannot be deleted). Note that if you try to delete a user-defined unit that is being used in an expression somewhere in your model, GoldSim will not allow you to do so, and will provide the following message:

Pressing the **Yes** displays a list of all of the elements which reference the unit.  

In some cases, you may want to create a new unit to track particular items (such as widgets, boxes, patients, etc.). Such a unit actually represents an *amount* of something. You could leave objects like these dimensionless, but for the purpose of viewing and plotting results, it may often be useful to assign units (e.g., 10 boxes instead of simply 10).

To do this, you can create a new dimensionless, as follows:

1. Create a new unit (e.g., widgets, with the abbreviation wdg)
2. When defining a value for the unit, define it as: 1 item.
3. Assign your new unit to the “Items” category.

Item is a built-in unit with no dimensions assigned to the “Items” category. The “Items” category also has several other built-in commonly required dimensionless units (*pers* for persons, *kpers* for thousands of persons, and *Mpers* for millions of persons).

You can repeat this process for other items (e.g., bolts, nuts, cars, etc.).

---

### Creating New Units for Items (Such as Widgets)

**Warning:** If you define two “item units” like this, they can be added together and their resulting units would be dimensionless. For example, if a bolt was defined as 1 item, and a nut was defined as 1 item, you could add 100 bolts to 200 nuts, and express the result in any dimensionless unit (e.g., pers).

---

### Managing User-Defined Units

There are two kinds of user-defined units:

- Units that are stored in your *system units file*, and hence are available to any model that you edit on your computer. The system units file (*units.dat*) is located in an application data subfolder on your computer (the location of this file differs depending on the operating system). That is, these units are stored separately on your computer (i.e., not just in the model file itself). These are highlighted in bold red font in the Units Manager, and are referred to here as system units.

- Units that are only stored within the model file itself, and are not available to any other models that you edit on your computer. These are highlighted in red (but not bold) in the Units Manager, and are referred to here as model units.
Any unit that you create on your computer automatically becomes a system unit (i.e., it is stored in the system units file, and hence is subsequently available to any other model you open on your computer.

Whenever a file is saved, any user-defined units that it uses are saved with the file. Therefore, if you open a file that contains user-defined units created by someone else, these units will be available in the file (as model units). They will not, however, become part of your system units file. That is, they do not become system units and are only local to that model. Hence, they would appear in the Units Manager when viewing that file, but would not be bold.

**Note:** If you open a model with a user-defined model unit, and the same unit is defined as a system unit on your computer (but has a different definition), GoldSim will display a warning message. The model unit will be ignored and the system unit will be applied for that model (which could cause some expressions to become invalid).

If you have opened a file with a model unit, and you would like to convert this to a system unit (so it is available to other models that you open or create), you can do so as follows:

1. Open the Units Manager.
2. Find the unit and click on it.
3. Click through the Wizard until you press Finish.
4. The unit will then become a system unit.

If you want to delete a system unit from your computer (so it is no longer available to other models you open or create), you can do so as follows:

1. Open the Units Manager.
2. Find the unit and select it.
3. Press the Remove button.
4. If the unit does not exist in the current file, it will immediately be removed from the Units Manager (and hence will no longer be a system unit). If the unit does exist in the current file, it will be converted from a system unit to a model unit (and hence will no longer be bold in the Units Manager).

When managing user-defined units, you must take care when copying portions of a model that utilize user-defined units between files:

- If you copy a portion of one model which references a user-defined unit to a second model which does not have that unit, you will need to create the unit in the second model (since otherwise it will not recognize the unit).

- If you copy a portion of one model which has a user-defined unit to a second model which has the same user-defined unit with a different definition, the definition of the unit in the second (destination) model will be used in the combined model, and this may result in an incorrect calculation or an invalid expression.

### Using Placeholders for Time Units

GoldSim provides two special units that act as “placeholders” or “keywords” when dealing with time units in elements. The keyword is replaced by an appropriate unit at runtime. These units are particularly useful if you plan to
change the Time display units or are switching back and forth between Elapsed Time and Calendar Time simulations.

**Read more:** Defining the Time Basis and Simulation Duration (page 453).

The two keywords are “datetime” and “date”. These can be entered as the display units (instead of an actual time unit) for an element. They cannot, however, be combined with other units in unit strings (e.g., to form strings like m3/date).

When running an Elapsed Time simulation, if “date” or “datetime” are the assigned display units for an element, the Time display units (specified in the Simulation Settings dialog) are used.

When running a Calendar Time simulation, if “date” or “datetime” are the assigned display units for an element, the value is displayed as a date (or a date and a time).

Note that when running a Calendar Time simulation, “date” or “datetime” should not be used to represent a duration or an amount of time. For example, the difference between two dates (“17 October 2009” – “1 October 2009”) represents an amount of time equal to 16 days. If such an expression was assigned units of “date”, it would display a Julian date as shown below (Julian dates are computed from 30 December 1899):

![Display Units: Date Type... Scalar](image)

Obviously, this would not be appropriate. By specifying the Display Units as days, the value is displayed as follows:

![Display Units: Day Type... Scalar](image)

**Read more:** Referencing Dates in Expressions (page 131).

GoldSim provides a special percentage unit (%). You can enter a number as a percentage, and GoldSim will automatically treat this as a fraction:

![Current Value = 0.1 10%](image)

You can also use this as a display unit, so that fractions can be displayed as percentages:

![Current Value = 60 %](image)

**The Options Dialog**

The Options dialog (accessed via Model|Options from the main menu) provides access to a number of options for controlling how GoldSim behaves.

There are always at least three tabs on the Options dialogs (some extension modules add additional tabs).

The **General** tab of the Options dialog provides a number of options for controlling the way that GoldSim behaves. These options are listed below.
Update expressions (F9) automatically after all moves and pastes: If this checkbox is checked (the default), GoldSim automatically updates the model (reconnecting links) whenever an element is moved or pasted. Otherwise, you must press F9 to update a model.

Show labels of elements with note attached in blue: If this checkbox is checked (the default), whenever a note is attached to an element, the element’s label (which acts as a hyperlink to accesses the note) will be blue. Otherwise, it will use the specified label color (which defaults to black).

Read more: Creating, Editing and Viewing Notes (page 755).

Save crash recovery information every X minutes: If this checkbox is checked (the default), GoldSim saves a copy of the model file periodically in order to allow you to recover a copy of your model file should GoldSim unexpectedly terminate for some reason while your file is open. This would allow you to restore changes you made to the file since you last manually saved it.

Read more: Restoring Files After an Unexpected Failure Using Auto-Save (page 63).

Simulation Events to record in Run Log: These checkboxes allow you to capture information about key events that may affect the behavior of your model. Their purpose is to help you to understand and diagnose your model’s behavior. The logging information identifies the exact time and nature of each of the selected events. The Run Log can be viewed via the main menu (Run | View Run Log) following a simulation. Note that once you have used the log information to identify elements of interest you may want to add Time History elements to track their behavior more closely, by monitoring all of their inputs and outputs.

Read more: The Run Log (page 548).

Reservoir and Pool element state changes: This option records when Reservoirs and Pools reach or drop below their upper bounds, and when they drop down to or rise above their lower bounds.

Read more: Defining Upper and Lower Bounds for a Reservoir (page 240); Defining Upper and Lower Bounds for a Pool (page 258).

Conditional Container activations and deactivations: This option records the times of activation and deactivation for all Conditional Containers, and in the case of deactivation events, indicates whether the Container’s Completion status was true when it was deactivated.

Read more: Activating a Container (page 904); Deactivating a Container (page 905).

Associate GSM files with this GoldSim: Oftentimes, you might have multiple versions of GoldSim on your computer. Windows associates the extension (.gsm) with the last version that was installed. If you double-click on a .gsm file, it opens it in the version with which that extension was associated. In some cases, however, you may want to ensure that the version that is currently open is the version that you want the .gsm file to be associated with. You can accomplish this by pressing the Associate button here.

The Graphic Tab of the Options Dialog

The Graphic tab of the Options dialog provides a number of options for controlling the way that graphics behave in GoldSim. These options are listed below.
Use saved container position and scale: If this checkbox is checked (the default), GoldSim remembers the scale and the location of a Container when you leave it. When you reenter the Container, it is displayed at the same scale and at the same position. Otherwise, once you leave the Container, all position and scale information is lost (it is reset back to the default – 100% and upper left-hand corner).

Use default line and polyline head: This is the default line head used when you add a line or polyline to the graphics pane.

Specify font used for new and pasted text: This is the font used when you add a text object to the graphics pane (by inserting a text object or pasting text from the clipboard).

Tabs in pasted text: These options allow you to control how text with tabs that is pasted into the graphics pane is treated. You can instruct GoldSim to delete the tabs, or replace them with spaces.

The Results Tab of the Options Dialog

The Results tab of the Options dialog provides a number of options for controlling the way that Results are displayed. The first two options control how numeric values are displayed in charts, tables and tool-tips:

Minimum number of significant figures to display: This allows you to control the number of significant figures displayed in result displays and tool-tips (as described below). This setting can be changed dynamically from within most result displays using Alt-Left and Alt-Right.

Use scientific notation if absolute value is >=: This allows you to control when scientific notation is used in result displays and tool-tips (as described below). This setting can be changed dynamically from within most result displays using Alt-Up and Alt-Down.

When displaying numeric values (other than dates), GoldSim respects the two settings noted above. Several points related to how these are applied should be noted:

- If the magnitude of the value is less than 0.0001 or greater than 1e10, GoldSim will always use scientific notation (rounding it at the prescribed number of significant figures). Otherwise, it will use the specified setting.

- If the magnitude of the value is below the threshold for scientific notation, GoldSim rounds off any decimal places beyond the prescribed number of significant figures and displays the value conventionally, with a decimal point if it has a fractional part. For example, to display 123.456 with four significant figures, GoldSim would show 123.5.

- When displaying values conventionally (i.e., not using scientific notation), GoldSim never rounds off left of the decimal point; hence the setting represents the minimum number of significant figures, not necessarily the actual number of significant figures (i.e., GoldSim may show more). For example, For example, to display 123456.7 with four significant figures specified, GoldSim would show 123457.

- When labeling chart axes, GoldSim respects the specified scientific notation setting, but ignores the significant figures setting (significant figures in chart axes are determined automatically and cannot be user-controlled).

- Values displayed in Result elements are limited to single precision (7 significant figures). Values in tool-tips can show up to 16 significant figures.
Other options on the Results tab are as follows:

**Show condition result outputs as**: This allows you to select how condition outputs are displayed in tables, tool-tips and input fields (e.g., 1/0, True/False, On/Off, etc.).

**Avoid scientific notation for currencies**. If this option is checked (the default), any value that represents a currency will never be displayed in any result using scientific notation. If it is cleared, the rules outlined above will be used.

**Automatic Export for Result Elements**. This option allows you to control how time histories are exported from Time History Result elements in your model. The options in the drop-list affect all Time History Result elements which are set to export automatically when the simulation completes. The default is “Export results after simulation”. If “Prompt before exporting results” is selected, at the end of the simulation, you will be prompted to determine whether or not to carry out an automatic export. If “Do not export result” is selected, no automatic export is carried out (i.e., this overrides the selection for each individual Result element). Pressing the **Export Now** button manually exports results from all Time History Result elements which are set to export automatically when the simulation completes.
Chapter 7: Running a Model

Computers are useless. They can only give you answers.

Pablo Picasso

Chapter Overview

After you have created a model, you need to run the model in order to produce results. This chapter describes how to control your simulation (e.g., specify its duration), specify the types of results you wish to save, and run the model.

In this Chapter

The following topics are discussed in this chapter:

- Simulation Settings
- Understanding and Referencing Run Properties
- Saving Outputs as Results
- Running and Viewing the Status of a Simulation
- Creating, Running and Comparing Scenarios
- Running an Optimization
- Running Sensitivity Analyses
- The Run Log
- Running GoldSim from the Command Line
Simulation Settings

GoldSim is a dynamic simulator, which means that your model can evolve and change with time. In order to carry out a dynamic simulation, GoldSim steps through time in discrete intervals (referred to as timesteps).

Calculations are carried out every timestep, with the values at the current timestep computed as a function of the values at the previous timestep. In GoldSim, you specify the duration of the simulation (e.g., 1 year) and the length of the timestep (e.g., 1 day).

The appropriate timestep length is a function of how rapidly the system represented by your model is changing: the more rapidly it is changing, the shorter the timestep required to accurately model the system. GoldSim allows you to change the timestep during a simulation (e.g., use short timesteps at early times when things are changing rapidly, and larger timesteps at later times).

To run a model, you must first specify its simulation settings. When you create a new model, GoldSim will display the Simulation Settings dialog. If you choose not to define the settings immediately, or if you wish to modify the settings subsequently, they can be accessed directly from the main menu under Run | Simulation Settings..., by pressing F2, or by clicking on the Simulation Settings button in the standard toolbar.

The simulation settings dialog is shown below:

The dialog consists of four tabs.

The Time tab is used to specify the time options for the simulation, such as the duration of the simulation, the length of the timestep, and the units in which time will be displayed in results.
The Monte Carlo tab is used to specify Monte Carlo options (how probabilistic simulations will be carried out), such as the number of realizations, and whether Latin Hypercube sampling is to be used.

The Globals tab is used to define global properties that can be referenced throughout your model.

The Information tab is used to specify the model author and a description for the simulation. This tab also provides some summary information regarding the model (e.g., the number of elements and the number of levels of containment).

Setting the Basic Time Options

The Time tab of the Simulation Settings dialog is used to specify the time options for the simulation:

There are four primary things that you specify from this dialog:

- The time basis and simulation duration;
- The timestep length and how the timesteps are aligned;
- Special reporting periods over which results can be accumulated and/or averaged; and
- When results will be saved.

This dialog also includes a button to Show Scheduled Updates…, which displays all of the model timesteps, and an Advanced button, for specifying a number of advanced time settings.

These items are discussed in detail in the sections below.

Defining the Time Basis and Simulation Duration

The top portion of the Time tab of the Simulation Settings dialog is used to specify the time basis for the simulation, as well as the simulation duration:
There are three ways in which you can represent time in GoldSim (controlled by **Time Basis** drop list):

- Elapsed Time
- Calendar Time
- Static Model

In an Elapsed Time simulation (the default), you must specify a simulation **Duration**. The Duration must be entered as a number (it cannot be defined as a link from an element). The simulation is then tracked in terms of elapsed time (e.g., when plotting time history results, the X-axis is plotted as elapsed time).

In a Calendar Time simulation (also referred to as a date-time simulation), you enter a **Start Time** and an **End Time**, and the simulation is then tracked in terms of the calendar time (e.g., when plotting time history results, the X-axis is plotted as dates/times).

If you select a Calendar Time simulation, you can click on the arrow in the box to view a calendar to assist you in selecting a start or end date:

You can select part of the date (month, day, year) in the input field and then enter a value or use the arrow keys to increment or decrement the value. Arrow buttons on the calendar move forward or backward one calendar month. Clicking the month on the calendar activates a menu with the twelve months. Clicking the year on the calendar activates a control for changing the year.

**Note:** GoldSim supports a date range from 1 January 1700 to 31 December 9999.

**Note:** The format in which the dates are displayed is determined by the Windows settings for your machine. To change these settings (e.g., to display dates in European format with the day before the month), go to Control Panel and edit the Regional Options.
Simulation Settings

Note: When running a Calendar Time simulation, you can also specify the Duration. When you do so, GoldSim will use the specified Start Time, and automatically update the End Time accordingly.

Whether you are running an Elapsed Time simulation, or a Calendar Time simulation, by referencing GoldSim’s built-in Run Properties, you can reference calendar-based information in your models (e.g., what day of the week is it? What month is it?). This is possible in Elapsed Time simulations also, since you can specify the Start Time for them.

Read more: Understanding and Referencing Run Properties (page 486).

Note: Specifying the Start Time for Elapsed Time simulations also allows you to enter time series data in terms of either elapsed time or calendar time; GoldSim uses the Start Time to convert between the two formats.

Read more: Entering Time Series Data as Dates or Elapsed Times (page 197).

The third option for carrying out your simulation is to run a Static Model. In a Static Model, the model does not step through time. This can be useful if you simply wish to carry out a static calculation using Monte Carlo simulation. A Static Model effectively sets the Duration to zero.

The Time Display Units provide the default time unit that is used in several parts of GoldSim. For example, for an Elapsed Time simulation, these are the units in which results (e.g., time history plots) will be displayed for all elements. Another example of where the default time unit is used is the Reservoir element. In this element, you specify a Display Unit, but several input fields require a rate. The default units for the rate are determined by the Time Display Units.

Example files which illustrate Elapsed and Calendar Time simulations (Elapsed.gsm and Calendar.gsm) can be found in the General Examples/Running folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

In order to carry out a dynamic simulation, GoldSim steps through time in discrete intervals (referred to as timesteps). Calculations (referred to as updates of the model) are carried out at end of every timestep, with the values at the end of the current timestep computed as a function of the values at the end of the previous timestep. Although the term timestep actually refers to an interval of time, it is often used interchangeably with the term update (indicating a calculation at a point in time).

In GoldSim, there are two kinds of updates/timesteps: scheduled updates (or timesteps) and unscheduled updates (or timesteps).

Scheduled Updates

Scheduled updates are specified directly prior to running the model. That is, you tell GoldSim when you want these updates to occur.

There are four ways to specify scheduled updates:

- You define a Basic Step of a specified length;
- You define Reporting Periods (e.g., monthly, annual) which are used to accumulate or average results, and which force a scheduled timestep at the end of each period;
- You define Capture Points, which represent specific times during the simulation where “snapshots” of statistical values can be viewed. A scheduled timestep is created for each Capture Point. By default, there is only a single Capture Point (the end of the simulation), but you can add as many as desired; and

- You define Period Timesteps for specific time intervals in which even smaller timesteps are desired.

**Read more:** [Specifying the Basic Step Length and Alignment](page 459); [Defining Reporting Periods](page 461); [Adding Shorter Timesteps Over Defined Periods](page 467); [Creating Capture Points for Final Value Results](page 469).

The most common way to create scheduled updates will be by specifying some combination of Basic Steps and/or Reporting Periods:

- **Basic Steps only.** In many cases, you will simply create a Basic Step (and this is the default). This is the simplest way to create scheduled updates (but is not necessarily the most effective way, depending on your application). You specify a Basic Step at which results are computed. You can subsequently choose to save results at each Basic Step, or at only some Basic Steps (e.g., every other Step, or every tenth Step).

  **Read more:** [Specifying When Time History Results Will Be Saved](page 464).

- **Reporting Periods only.** In many cases, you will not use a Basic Step at all, but will simply define your scheduled updates using Reporting Periods. Why would you do so? Whereas results saved at Basic Steps always represent instantaneous values, results saved at Reporting Periods can not only be displayed as instantaneous results, but can also report accumulated or averaged values over a specified period (e.g., monthly). Hence, if you needed average or accumulated flows over each step (e.g., over the last day or the last month), you can do so by defining your steps using Reporting Periods.

  **Note:** Most spreadsheet models implicitly compute *accumulated* values (e.g., flows) over a step (or the change from one step to the next). Hence, if you want to compare or integrate GoldSim with a spreadsheet model, you should use Reporting Periods to define your steps.

- **Reporting Periods and Basic Steps.** In some cases, you may want to create Reporting Periods in order to report accumulated or averaged values over a specified period, but you may also need a shorter computational step in order to ensure accuracy. For example, you may require a daily timestep to ensure accuracy, but only need to view monthly averages or accumulated values in terms of results. In this case, you would create a Basic Step as well as Reporting Periods. You could choose to save results only for Reporting Periods, or could also save results for the Basic Steps. When defining both Basic Steps and Reporting Periods, Reporting periods can be defined to coincide with Basic Steps (e.g., a 1 day Basic Step and a monthly reporting period), but they do not have to. However, if a Basic Step is specified, Reporting periods must always be at least twice as large as the Basic Step.
In some cases, you may also want to create Capture Points, which are typically used to capture and view statistical results at specified points in your simulation. In particular, Capture Points allow you to save and view “Final Value” type results at specified times during a simulation (rather than just at the final time point). Final Value results are associated with multiple realizations (or items in an array), and allow you to display distributions, multi-variate results (e.g., scatter plots) and array plots.

**Read more:** [Viewing the Four Basic Result Types](#) (page 558).

When you define a Capture Point, a scheduled timestep is added at that time (if one does not already exist).

Finally, in rare cases, you may want to create smaller timesteps over a defined period of your simulation (e.g., if you know that early in your simulation, variables will change rapidly, and hence a small timestep is required).

Note: You can also specify a shorter timestep for different parts (i.e., Containers) of your model.

**Read more:** [Specifying Containers with Internal Clocks](#) (page 474).

When specifying scheduled timesteps, the appropriate timestep length is a function of how rapidly the system represented by your model is changing: the more rapidly it is changing, the shorter the timestep required to accurately model the system.

You can view all of the scheduled timesteps in your simulation by pressing the **Show Scheduled Updates** button at the top of the **Time** tab. The following dialog will be displayed:
Simulation Settings

This provides a summary of the scheduled timesteps you have defined (with the final column listing the type of each time point).

**Unscheduled Updates**

In some cases, events or other changes in the model may not fall exactly on a scheduled update. That is, some events or changes may actually occur between scheduled updates of the model. These trigger an “unscheduled update” of the model. Unscheduled updates are timesteps that are dynamically inserted by GoldSim during the simulation in order to more accurately simulate the system. That is, they are not specified directly prior to running the model. GoldSim inserts them automatically (and, generally, without you needing to be aware of it).

“Unscheduled updates” can be generated in the following ways:

- When events are output by a Timed Event, Event Delay, Discrete Change Delay or Time Series element;
- By manually specifying a dynamic timestep (i.e., dynamically controlling the time between updates);
- When a Reservoir or Pool element reaches an upper or lower bound;
- When a Resource becomes exhausted;
- When any element is triggered by an At Stock Test, At Date, At Etime or At Duration triggering event; and

Read more: Understanding Event Triggering (page 355); Timed Event Elements (page 365); Event Delay Elements (page 377); Delaying a Discrete Change Signal (page 390); Generating Discrete Changes Using Time Series Elements (page 415); Dynamically Controlling the Timestep (page 471); Defining Upper and Lower Bounds for a Reservoir (page 240); Defining Upper and Lower Bounds for a Pool (page 258); Using Resources (page 839).

When any of these events occur, GoldSim automatically inserts an unscheduled update at the exact time that the event or change occurs. For example, if you had specified a one day timestep, and a Timed Event occurs at 33.65 days (i.e., between the scheduled one-day updates), GoldSim would insert an unscheduled update at 33.65 days.

**Note:** By default, GoldSim always inserts unscheduled updates. However, although it is generally not recommended, you can override the default and instruct GoldSim not to insert the unscheduled updates. (This option is accessed via the Advanced… button in the Time tab of the Simulation Setting dialog).

Read more: Controlling Unscheduled Updates (page 470).

A key and important difference between scheduled updates and unscheduled updates is that scheduled updates are included in time history plots and tables (unless you choose to exclude them). Unscheduled updates, however, do not normally appear in time history plots and tables. That is, although these timesteps may affect the results (e.g., by making them more accurate at the
scheduled timesteps), unscheduled updates of the model are not saved and plotted. Only the scheduled updates are actually saved and plotted.

**Note:** In some cases, it may be of interest to see the values of selected outputs at unscheduled updates. To facilitate this, GoldSim provides an option to do so (under a specified set of conditions) in the Advanced Time settings.

**Read more:** Including Unscheduled Updates in Time History Results (page 477).

A simple example file illustrating unscheduled updates (UnscheduledTimeSteps.gsm) can be found in the Running subfolder of the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

The GoldSim timestepping algorithm is discussed in detail in Appendix F.

The middle portion of the Time tab of the Simulation Settings dialog is used to specify the Basic Step length and the manner in which the timesteps are aligned:

<table>
<thead>
<tr>
<th>Timestep Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment: Start Time aligned</td>
</tr>
<tr>
<td>Basic Step: User-specified 1 day</td>
</tr>
</tbody>
</table>

For many models, you will simply create a Basic Step of a specified length, and these will be the only scheduled timesteps you will need to define.

**Read more:** Understanding Timestepping in GoldSim (page 455).

The Basic Step field defines the length of the timestep. The Alignment field determines how the steps are aligned.

The manner in which these are specified differs based on whether the Time Basis is Elapsed Time or Calendar time.

**Read more:** Defining the Time Basis and Simulation Duration (page 453).

For an Elapsed Time simulation, the Basic Step is either “User-specified” or “No basic step”. If the latter is selected, you will need to specify Reporting Periods in order to step through the model (i.e., if no Basic Step is defined, Reporting Periods must be defined).

**Read more:** Defining Reporting Periods (page 461).

If “User-specified” is selected, you define a timestep length using any time unit. This can be specified as a number or an equation (but you cannot link to another element). For Elapsed Time simulations, the Alignment is not applicable (it is always “Start-time aligned” indicating that the steps start at an elapsed time of zero).

For a Calendar Time simulation, there are multiple options for the Basic Step:
As is the case for an Elapsed Time simulation, if “No basic step” is selected, you will need to specify Reporting Periods in order to step through the model (i.e., if no Basic Step is defined, Reporting Periods must be defined).

If “User-specified” is selected, you define a timestep length using any time unit. This can be specified as a number or an equation (but you cannot link to another element).

“Calendar years”, “Calendar quarters” and “Calendar months” represent the duration of the actual underlying calendar, and hence define a variable-length timestep. A spin-control can be used to specify the number of calendar units per step. For example, if you selected “Calendar months” and specified a value of 1, then the length of the timestep would vary between 28 and 31 days, depending on the month.

“Weeks”, “Days”, “Hours” and “Seconds” provide a simple way to define a fixed-length timestep. A spin-control can be used to specify the number of fixed-length units per step.

For Calendar time simulations, the **Alignment** determines how the timesteps are aligned relative to the **Start Time** of the simulation. There are two options: “Calendar aligned” and “Start Time aligned”. If “Start Time aligned” is selected, all timesteps are aligned with the start of the simulation. If “Calendar aligned” is selected, GoldSim uses the start of the selected calendar period to align the timesteps. To understand this, it is easiest to consider an example.

If you specified a **Basic Step** of “Calendar months” (and set it to 1), specified an **Alignment** of “Start Time aligned”, and a **Start Time** of January 15, then GoldSim would create timesteps at January 15, February 15, March 15, April 15, etc. On the other hand, if you specified a **Basic Step** of “Calendar months” (and set it to 1), specified an **Alignment** of “Calendar aligned”, and a **Start Time** of January 15, then GoldSim would create timesteps at January 15, February 1, March 1, April 1, etc.

Several points should be noted for Calendar Time simulations:

- The **Start Time** and the **End Time** are always included as scheduled timesteps, regardless of the Basic Step or Alignment.
- When using “Weeks” as the Basic Step, the first day of the week can be specified by the user. This is controlled via the Advanced… button at the bottom of the **Time** tab.

Read more: [Controlling When Weeks and Years Start](page 478).
Note: If you are unsure how a particular combination of Basic Step, Alignment and Start Time will affect the scheduled timesteps, simply press the **Show Scheduled Updates**… button at the top of the **Time** tab. This will display all of the scheduled timesteps prior to actually running the model. By experimenting with the various options and viewing the resulting scheduled timesteps, you should be able to quickly create the timestepping schedule that you require.

Note: The maximum number of Basic Steps that GoldSim will support is one million.

If you are saving a large number of time history results and have a large number of Basic Steps, you may not wish to save values at every Basic Step (as this could require a large amount of disk space). In fact, if you have defined Reporting Periods, you may only want to save results at those times (and may not need to save results at Basic Steps at all). To facilitate this, at the bottom of the **Time** tab of the Simulation Settings dialog, GoldSim allows you to specify when results are to be saved (e.g., every other Basic Step, every Basic Step, Reporting Periods only).

**Read more:** [Specifying When Time History Results Will Be Saved](#) (page 464).

Example files which illustrate Elapsed and Calendar Time simulations (Elapsed.gsm and Calendar.gsm) can be found in the General Examples/Running folder in your GoldSim directory (accessed by selecting **File | Open Example**... from the main menu).

In some cases, you need to compute and report *accumulated*, *average*, the *change* or the *rate of change* of values over specified periods (e.g., monthly, annually). For example, you may need to report the cumulative amount of money or water that moved from one point to another each month.

To support this, GoldSim allows you to define Reporting Periods. Scheduled updates (timesteps) are created at the end of each Reporting Period. Hence, these can also be thought of as “Reporting Steps”. Whereas results saved at Basic Steps always represent instantaneous values, results saved at Reporting Periods can not only be displayed as instantaneous results, but can also report accumulated, averaged, the change or the rate of change of values over a specified period (e.g., monthly).

**Read more:** [Understanding Timestepping in GoldSim](#) (page 455).

Reporting Periods are defined toward the bottom of the **Time** tab of the Simulation Settings dialog:

![Reporting Periods](#)

There are three options for Reporting Periods (referred to in the dialog explicity as Reporting Steps to remind you that these actually create scheduled updates): “None” (the default), “Major” (in which your define a single Reporting Period), and “Major & Minor” (in which you define two Reporting Periods).

The manner in which Reporting Periods are specified differs based on whether the **Time Basis** is Elapsed Time or Calendar time.

**Read more:** [Defining the Time Basis and Simulation Duration](#) (page 453).
For an Elapsed Time simulation, you simply specify the length of the Reporting Period for the Major and Minor Period:

![Simulation Settings](image)

*In this example, a Major Period would be created every 100 days from the start of the simulation; a Minor Period would be created every 10 days from the start of the simulation.*

You define the length using any time unit. This can be specified as a number or an equation (but you cannot link to another element).

**Note:** The Major Period must be a multiple of the Minor Period.

You must also define a **Period Label** for each type of period (which defaults to Major and Minor). These IDs are referenced when displaying Reporting Period results.

For a Calendar Time simulation, there are five options for the Major Period:

![Calendar Time Options](image)

These create Reporting Periods every year, quarter, month, week or day, respectively.

Depending on what you select for the Major Period, limited options will be valid for the Minor Period, since the Major Period must be a multiple of the Minor Period. For example, if you select “annual” for the Major Period, only three options are valid for the Minor Period (quarterly, monthly and daily). If you were to pick “monthly” for the Major Period, there is only one valid option for the Minor Period: daily. Weekly is never a valid option (and is not provided in the drop-list) for the Minor Period, because it does not divide evenly into any of the possible Major Periods.

Calendar Periods are placed according to the following rules:

- Monthly Periods are added on start of the first day of the calendar month (at midnight).
- Daily Periods are added at midnight on each day.
- Where Annual and Quarterly Periods are added is controlled by how you define the “Start of reporting year” which is controlled via the **Advanced…** button at the bottom of the **Time** tab. Annual Periods are added on the first day of the month selected as the “Start of the reporting year”. Quarterly Periods are added on the first day of the month selected as the “Start of the reporting year”, and the first days of the months 3, 6 and 9 months after the selected month.
- Where Weekly Periods are added is controlled by how you define the “First day of model week” which is controlled via the **Advanced…** button at the bottom of the **Time** tab. Weekly Periods are added at midnight of the start of the specified first day of the week.

**Read more:** [Controlling When Weeks and Years Start](page 478).
It is important to understand that in many cases, you should define your scheduled updates using Reporting Periods (and not bother to use a Basic Step at all). Recall that whereas results saved at Basic Steps always represent instantaneous values, results saved at Reporting Periods can not only be displayed as instantaneous results, but can also report the change, rate of change, accumulated or averaged values over a specified period (e.g., monthly). Hence, if you needed the change, rate of change, average or accumulated flows over steps (e.g., over the last day or the last month), you can do so by defining your steps using Reporting Periods.

Note: Most spreadsheet models implicitly compute accumulated values (e.g., flows) over a step (or the change from one step to the next). Hence, if you want to compare or integrate GoldSim with a spreadsheet model, you should use Reporting Periods to define your steps.

Of course, in some cases, you may want to create Reporting Periods in order to report accumulated or averaged values over a specified period, but you also may need a shorter computational step in order to ensure accuracy. For example, you may require a daily timestep to ensure accuracy, but only need to view monthly averages or accumulated values in terms of results. In this case, you would create a Basic Step as well as Reporting Periods.

When defining both Basic Steps and Reporting Periods, Reporting periods can be defined to coincide with Basic Steps (e.g., a 1 day Basic Step and a monthly reporting period), but they do not have to. However, if a Basic Step is specified, Reporting periods must always be at least twice as large as the Basic Step.

Once you have defined Reporting Periods, how can they be used? Reporting Periods impact your model in three ways:

1. Reporting Periods automatically create a scheduled timestep for your model (regardless of how you have defined your Basic Step).

   Read more: Understanding Timestepping in GoldSim (page 455).

   Note: If you press the Show Scheduled Updates... button at the top of the Time tab, all of the scheduled timesteps in the model will be displayed. Because Reporting Periods add scheduled timesteps, you can use this to see exactly where they will be added. Updates associated with Major and/or Minor Periods are specifically identified (since Minor Periods must divide evenly into Major Periods; any Major Period listed automatically also represents a Minor Period).

2. Results based on Reporting Periods (e.g., cumulative values over each period, average values over each period, etc.) can be accessed and viewed via Time History Result elements.

   Read more: Viewing Reporting Period-Based Results in Time History Result Elements (page 607).

3. When Reporting Periods are defined, a number of additional Run Properties are created that can be referenced in your model (e.g., the elapsed time from the start of the last Reporting Period, the remaining time to the start of the next Reporting Period).
A simple example file illustrating Reporting Periods (ReportingPeriods.gsm) can be found in the Running subfolder of the the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

If you are using Reporting Periods, time history results are always saved at the end of each reporting period. (Results are also always saved at the beginning and end of the simulation).

Note, however, that if you are using Basic Steps, you may not need to save values at every Basic Step (as this could require a large amount of disk space). To address this, GoldSim allows you to save only selected plot points.

At the bottom of the Time tab of the Simulation Settings dialog, GoldSim allows you to specify when time history results will be saved. If you only have defined Reporting Periods (and have not defined a Basic Step), this section of the dialog looks like this:

If you have only defined a Basic Step (and have not defined Reporting Periods), it looks like this:

If you have defined both a Basic Step and Reporting Periods, you can select to save both Reporting and Basic Steps:

Whenever Reporting Periods are defined (as in the first and third cases), results will automatically be saved at the end of every Reporting Period (as well as at the beginning and end of the simulation). In fact, in the first case (only Reporting Periods), you have no options for specifying when results are to be saved.

Whenever a Basic Step is defined, however, you have the option of specifying how often you want to save Basic Steps. If Save Every is entered as 1, every Basic timestep is saved; if it is specified as 2, every other timestep is saved; if it is specified as 5, every fifth timestep is saved, and so on. If it is specified as 0 (the default), no Basic Steps are saved.

Note: The discussion above is only associated with when time history results are saved. It does not control which time history results are saved. Whether or not time history results are saved for a specific element is a function of 1) whether the simulation has multiple realizations, 2) the status of the check box for the element defining if Time History results are to be saved; and 3) whether or not the element is referenced by a Time History Result element.

Read more: Saving Outputs as Results (page 494).
**Note:** As an advanced timestepping option, GoldSim allows you to define shorter timesteps over a defined period. For example, if you know that early in your simulation, variables will change rapidly, you may want to use a small timestep initially. To support this, GoldSim allows you to select periods during your simulation in which the timestep is decreased for a specified duration, before returning to the timestep defined via the Basic Step (or Reporting Period). When this is done, the **Save Every** option applies to these shorter steps. For example, if you have a 100 day simulation, a 5 day basic step, a 1 day step applied for the first 10 days, and **Save Every** is set to 2, then GoldSim would save time history results at 0, 2, 4, 6, 8, 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100 days.

*Read more:* [Adding Shorter Timesteps Over Defined Periods](page 467).

Based on the number of Basic Steps (and how often they are to be saved), the number of Reporting Periods, the number of elements for which time histories are being saved, the number of realizations, and several other settings (e.g., whether or not some advanced timestepping options are being used), GoldSim reports the amount of disk space required to store the results at the bottom of the **Time** tab. This is also provided in the **Information** tab of the Simulation Settings dialog and the **Information** tab of each Container.

*Read more:* [Viewing and Editing Model Summary Information](page 485); [Summary Information for a Container](page 140).

If you press the **Show Scheduled Updates…** button at the top of the **Time** tab, all of the scheduled timesteps in the model will be displayed:

The dialog not only indicates when the scheduled timesteps will occur, but it also indicates which of these steps will be saved for time history display. In the example above, it indicates that every fifth Basic Step is being saved for display.
Note: In some cases, it may be of interest to display the values of selected outputs that were computed not only at scheduled updates (e.g., Basic Steps and Reporting Periods), but also at unscheduled updates (e.g., when random events occur). To facilitate this, GoldSim provides an option to do so (under a specified set of conditions) in the Advanced Time settings. When you display results that show unscheduled updates, all Basic Steps are also displayed (i.e., for these specific displays, the Save Every option does not apply).

Read more: Understanding Timestepping in GoldSim (page 455); Including Unscheduled Updates in Time History Results (page 477).

The Advanced… button in the Time tab of the Simulation Settings dialog provides access to a dialog for specifying a number of advanced timestepping features:

This dialog is used to specify several different things:

- Adding shorter (magnified) timesteps over specified periods;
- Adding Capture Points during a simulation, at which results are captured in order to display distributions, multi-variate results (e.g., scatter plots) and arrays at specified points in time.
- Controlling unscheduled updates;
- Dynamically controlling the timestep during a simulation;
- Including unscheduled updates in time history results;
- Referencing and displaying Time when editing a model; and
• Defining the first day of the week and first month of the year. These are discussed in detail in the sections below.

Note that the settings shown above are the default settings. If any of the default settings are changed, GoldSim indicates this in the Time tab of the Simulation Settings dialog by placing a check on the Advanced… button:

In some cases, you may want to create smaller timesteps over a defined period of your simulation. For example, if you know that early in your simulation, variables will change rapidly, you may want to use a small timestep initially. To support this, GoldSim allows you to select periods during your simulation in which the timestep is decreased for a specified duration, before returning to the timestep defined via the Basic Step (or Reporting Period).

These periods during which the timestep is shorter are defined at the top of the Advanced Time Settings page (accessed via the Advanced… button on the Time tab of the Simulation Settings dialog):

You can add a new time period with a shorter timestep by pressing the Add… button. When you do so, a dialog for specifying the period will be displayed. The dialog differs based on whether the Time Basis is Elapsed Time or Calendar time.

Read more: Defining the Time Basis and Simulation Duration (page 453).

If you are carrying out an Elapsed Time simulation, the dialog looks like this:

The elapsed Start Time and End cannot overlap with another defined period. The Period Timestep must divide evenly into the Basic Step (if one is defined), or the shortest Reporting Period (if there is no Basic Step). It must be smaller than the period it is dividing. The Start Time, End Time and Period Timestep can be defined using any time unit. This can be specified as a number or an equation (but you cannot link to an element).

If you are carrying out a Calendar Time simulation, the dialog looks like this:
The **Start Time** and **End Time** are specified as dates/times cannot overlap with another defined period. The **Period Timestep** must divide evenly into the Basic Step (if one is defined), or the shortest Reporting Period (if there is no Basic Step). If the Basic Step is defined as a calendar duration (e.g., Calendar months), then the **Period Timestep** must divide evenly into a day. The **Period Timestep** must be smaller than the period it is dividing. It can be defined using any time unit. This can be specified as a number or an equation (but you cannot link to an element).

**Note:** If you press the **Show Scheduled Updates** button at the top of the **Time** tab, all of the scheduled timesteps in the model will be displayed. Because Period Timesteps add scheduled timesteps, you can use this to see exactly where they will be added.

It is important to understand how GoldSim actually schedules these Period Timesteps, and this can best be done by considering an example. Consider a 100 day simulation with a 20 day basic timestep. By default, scheduled updates would then occur at 0, 20, 40, 60, 80 and 100 days. Now assume that we schedule a Period Timestep that looks like this:

When would the updates occur? In this case, they would occur at 0, 20, 28, 32, 36, 40, 60, 80 and 100 days. Note that there is not a step at 25 days. The first shortened step occurs at 28 days. GoldSim simply adds timesteps at all multiples of the Period Timestep (in this case 4 days) that fall within the Period (in this case 25 to 40 days). The first multiple of 4 that falls within the Period is 28.

By default, Period Timesteps are not saved as part of time history results. However, if you check **Include in Time Histories**, these scheduled timesteps will be saved. In this case, the **Save Every** option on the main **Time** tab applies to these shorter steps. For example, if you have a 100 day simulation, a 5 day basic step, a 1 day step applied for the first 10 days, and **Save Every** is set to 2, then GoldSim would save time history results at 0, 2, 4, 6, 8, 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100 days.

**Read more:** [Specifying When Time History Results Will Be Saved](page 464).

Once you have added a Period Timestep, you can edit it by pressing the **Edit** button, or delete it by pressing the **Remove** button.
**Note:** In addition to allowing you to create a predefined schedule of timestep changes using Period Timesteps, you can also instruct GoldSim to dynamically adjust the timestep during a simulation based on the values of specified parameters in your model, or adjust the timestep locally within different portions of your model.

**Read more:** [Dynamically Controlling the Timestep](page 471); [Specifying Containers with Internal Clocks](page 474).

A simple example file illustrating adding shorter timesteps over defined periods (PeriodsWithShorterSteps.gsm) can be found in the Running subfolder of the the General Examples folder in your GoldSim directory (accessed by selecting **File | Open Example**... from the main menu).

When viewing results at the end of a simulation, you can view two types of results: Time Histories or Final Values. Final Value results are usually associated with multiple realizations (or items in an array), and allow you to display distributions, multi-variate results (e.g., scatter plots) and array plots.

**Read more:** [Viewing the Four Basic Result Types](page 558).

By default, “Final Value” type results are only available at the final time point in the simulation. In some cases, however, you may also want to capture these results at other times in the simulation. This can be useful, for example, if you wanted to view a result distribution for an output at various times during a simulation. GoldSim facilitates this by allowing you to create Capture Points at which these results are also made available.

Capture Points are defined toward the top of the Advanced Time Settings page (accessed via the **Advanced**... button on the **Time** tab of the Simulation Settings dialog):

![Advanced Time Settings](image)

Note that there is always a Capture Point at the end of the simulation (which cannot be deleted). You can add a new Capture Point by pressing the **Add**... button. When you do so, a dialog for specifying the Capture Point will be displayed. The dialog differs based on whether the Time Basis is Elapsed Time or Calendar time.

**Read more:** [Defining the Time Basis and Simulation Duration](page 453).

If you are carrying out an Elapsed Time simulation, the dialog looks like this:
The elapsed Capture Time should be within the specified simulation duration, and can be defined using any time unit. This can be specified as a number or an equation (but you cannot link to an element).

If you are carrying out a Calendar Time simulation, the dialog looks like this:

![Capture Point Dialog](image)

The Capture Time is specified as a date/time, and should be within the specified simulation period.

When creating a Capture Point, you must provide it with an ID (which can be up to 20 characters long). This ID is used as a label when accessing Capture Point results in various result displays.

**Read more:** Viewing Results at Capture Points (page 569).

Once you have added a Capture Point, you can edit it by pressing the **Edit** button, or delete it by pressing the **Remove** button.

When you define a Capture Point, a scheduled timestep is added at that time (if one does not already exist).

**Note:** If you press the **Show Scheduled Updates…** button at the top of the **Time** tab, all of the scheduled timesteps in the model will be displayed. Because Capture Points add scheduled timesteps, you can use this to see exactly where they will be added. Note, however, that although Capture Points are scheduled timesteps, they are not included in time history results.

A simple example file illustrating Capture Points (CapturePoints.gsm) can be found in the Running subfolder of the General Examples folder in your GoldSim directory (accessed by selecting **File | Open Example...** from the main menu).

**Controlling Unscheduled Updates**

In order to carry out a dynamic simulation, GoldSim steps through time in discrete intervals (referred to as timesteps). Calculations (referred to as updates of the model) are carried out at end of every timestep. Although the term timestep actually refers to an interval of time, it is often used interchangeably with the term update (indicating a calculation at a point in time). In GoldSim, there are two kinds of updates/timesteps: scheduled updates (or timesteps) and unscheduled updates (or timesteps).

**Read more:** Understanding Timestepping in GoldSim (page 455).

Scheduled updates are specified directly prior to running the model. That is, you tell GoldSim when you want these updates to occur. Unscheduled updates are timesteps that are dynamically inserted by GoldSim during the simulation in
order to more accurately simulate the system. That is, they are not specified directly prior to running the model. GoldSim inserts them automatically (and, generally, without you needing to be aware of it).

By default, scheduled updates are always dynamically inserted by GoldSim. However, in some (rare) cases, you may want to prevent unscheduled updates from being inserted. For example, if your model included a specialized algorithm that was designed based on the assumption that the timestep was constant, inserting unscheduled updates could invalidate the algorithm.

To support such situations, GoldSim allows you to disable unscheduled updates. You can do so by clearing the checkbox labeled **Allow unscheduled updates (recommended):**

![Advanced Time Settings](image)

**Warning:** Because unscheduled updates are intended to more accurately represent a complex dynamic system, disabling this feature should be done with caution, and is generally not recommended. In most cases, it will have the effect of deferring events to the next scheduled update, which under some circumstances could cause significant inaccuracies. In some cases (e.g., a Reservoir or Pool hitting an upper bound), its effects can be somewhat more complex (e.g., it changes how an overflow rate is computed). Some advanced features in GoldSim cannot function properly at all without using unscheduled updates. In these situations, GoldSim will throw a fatal error during a simulation if you have disabled unscheduled updates and are using such a feature.

**Read more:** [How a Reservoir Computes the Overflow Rate](page 242).

GoldSim allows you to decrease the timestep length according to a specified schedule during a simulation (e.g., start with a small timestep, and then telescope out to a larger timestep) by defining time periods which have different timestep lengths.

**Read more:** [Adding Shorter Timesteps Over Defined Periods](page 467).

Although defining such time periods can be very useful, you must fully specify them prior to running the simulation. That is, you must know how you would like to alter your timesteps prior to running the model. In some cases, however,
it may not be possible to do this. That is, in complex systems (particularly ones with uncertain parameters), variables may change at different rates in different realizations, in ways that you cannot predict prior to running the model.

To better simulate these kinds of systems, GoldSim provides an advanced feature that allows you to dynamically adjust the timestep during a simulation (i.e., insert unscheduled updates) based on the values of specified parameters in your model. For example, you could instruct GoldSim to use a maximum timestep of 1 day if X was greater than Y, and 10 days if X was less than or equal to Y. Similarly, you could instruct GoldSim to use a short timestep for a period of 10 days after a particular event occurs, and then return to the default timestep.

This feature can be accessed in the Advanced Time Settings dialog (accessed from the Advanced… button on the Time tab of the Simulation Settings dialog). Within this dialog is a checkbox labeled Dynamically control time between updates:

By default, this checkbox is cleared. Checking it allows you to edit the Maximum time between updates field, which can be used to dynamically control the timestep (by inserting unscheduled updates).

Note: In order to access this checkbox, Allow unscheduled updates must also be checked (the default).

After GoldSim completes all the calculations for a timestep, it must decide when to insert the next timestep. Normally, the next timestep is specified by the Basic Step (or Reporting Periods or Period Timesteps, if defined).

GoldSim uses the value in the Maximum time between updates field (referred to as DT_{max} below) to override the normally scheduled timestep according to the following rules:

1. If DT_{max} is greater than or equal to DT_{sched} (the length of time until the next scheduled timestep), the field is ignored and the next timestep is simply the regularly scheduled timestep (T_{curr} + DT_{sched}), where T_{curr} is the current time.
2. If DT_{max} is less than or equal to (DT_{sched} / 2), the next timestep is T_{curr} + DT_{max}.
3. If DT_{max} is greater than (DT_{sched} / 2), but less than DT_{sched}, the next timestep is T_{curr} + (DT_{sched} / 2) – that is, DT_{sched} is divided into two equal halves.

These rules can be best understood by considering several simple examples:

**Example 1:** Assume that the next scheduled timestep length is 10 days, and you have specified a Maximum time between updates (DT_{max}) of 4 days. GoldSim would compute timesteps at the following points in the first 10 days:

- When time = 0 day, DT_{sched} = 10 days, so DT_{max} is less than (DT_{sched} / 2), and according to rule 2, the next step would be computed at 4 days.
• At time = 4 days, DT_{sched} = 6 days, so DT_{max} is greater than (DT_{sched}/2), but less than DT_{sched}, and according to rule 3, the next step would be computed at 7 days.

• At time = 7 days, DT_{sched} = 3 days, so DT_{max} is greater than DT_{sched}, and according to rule 1, the next step would be computed at 10 days.

**Example 2:** Assume that the next scheduled timestep length is 10 days, and you have specified a Maximum time between updates (DT_{max}) of 6 days. GoldSim would compute timesteps at the following points in the first 10 days:

• When time = 0 day, DT_{sched} = 10 days, so DT_{max} is greater than (DT_{sched}/2), but less than DT_{sched}, and according to rule 3, the next step would be computed at 5 days.

• At time = 5 days, DT_{sched} = 5 days, so DT_{max} is greater than DT_{sched}, and according to rule 1, the next step would be computed at 10 days.

**Example 3:** Assume that the next scheduled timestep length is 10 days, and you have specified a Maximum time between updates of 1 day. GoldSim would compute timesteps at the following points in the first 10 days:

• When time is equal to 0 through 8 days, rule 2 applies, and a timestep of 1 day is used (steps occur at 1, 2, 3, 4, 5, 6, 7, 8, and 9 days).

• At time = 9 days, DT_{sched} = 1 day, so DT_{max} is equal to DT_{sched}, and according to rule 1, the next step would be computed at 10 days.

In all three of these examples, the actual timestep length is always less than or equal to the specified Maximum time between updates. Moreover, the scheduled timesteps defined by the Basic Step, Reporting Periods or Period Timesteps are always observed.

**Note:** Because certain events (e.g., triggered by a Timed Event or a Reservoir hitting an upper bound) are automatically represented by GoldSim with greater accuracy by allowing them to fall between scheduled timesteps, any events that occur could result in a smaller timestep than that computed using the rules presented above. That is, after GoldSim computes the next timestep based on the rules presented above, it then checks to see if an event would fall within that interval. If an event does fall into that interval, the timestep could be shortened in order to represent the event.

**Read more:** [Understanding Timestepping in GoldSim](page 455).

In practice, you would not normally specify the Maximum time between updates as a constant (as you could accomplish the same thing by simply appropriately defining scheduled timesteps. Instead, you would specify it as a variable (i.e., a function). For example:

• If you wanted to reduce the timestep of a model from 10 days to 1 day whenever variable X was close to (say within 10% of) variable Y, you would define your Basic Step as 10 days, and define Maximum time between updates as the function: if(abs(X-Y)/Y < 10%, 1 day, 10 day).

• If you wanted to reduce the timestep to 1 day for a period of 20 days after a particular event occurred, and then return the timestep to its default (e.g., 10 days), you could do this by: 1) triggering an Event
Delay when the original Event occurs, with a delay time of 20 days; and 2) defining **Maximum time between updates** as the function:

\[ \text{if(EventDelay.Num\_in\_Transit > 0, 1 day, 10 day)}. \]

In addition to providing this dynamic timestepping algorithm on a global scale (i.e., for the entire model), GoldSim also enables you to apply a Maximum time between updates to specific Containers. As a result, you could apply a maximum timestep length only to a portion of your model (represented by a particular Container).

**Read more:** Specifying Containers with Internal Clocks (page 474).

---

**Note:** If the **Maximum time between updates** is less than or equal to zero, a warning message will be written to the Run Log, and it will be ignored (which is equivalent to clearing the **Dynamically control time between updates** checkbox).

**Note:** By default, time history displays only include scheduled timesteps, and will not capture unscheduled updates. However, GoldSim provides an option (under a specified set of conditions) in the Advanced Time settings that allows you to include both scheduled and unscheduled updates in a time history displays. This provides a mechanism to display and view the exact times at which unscheduled updates are inserted.

**Read more:** Including Unscheduled Updates in Time History Results (page 477).

---

In some situations, you may wish to use different timesteps in different portions of your model. This is useful, for example, if the processes represented in one portion of your model occur rapidly and hence require a short timestep to model accurately, while processes represented in the rest of your model occur much more slowly, such that a larger timestep can be used.

You can specify smaller timesteps for a Container by assigning the Container an “internal clock”. Containers can be assigned internal clocks by selecting the **Internal Clock** feature in the Container dialog. When you assign an internal clock to a Container, an **Internal Clock** tab is added to the Container dialog:
Containers with internal clocks are represented in the graphics pane as follows:

Note: When you specify a Container as having an **Internal Clock**, you cannot also define it as having **Looping Capability** (these two options are mutually exclusive).

Certain events (e.g. triggered by a Timed Event or a Reservoir hitting an upper bound) are automatically represented by GoldSim with greater accuracy by allowing them to fall between scheduled timesteps.

**Read more:** [Understanding Timestepping in GoldSim](#) (page 455).

An important implication of an Internal Clock Container is that any unscheduled timesteps generated inside the Container only interrupt the clock pertaining to that Container (and hence cause the elements in the Container to update). They do not cause elements outside the Container to be updated.

To understand this, consider the following simple example. Suppose that your model had a 10 day timestep. Within an Internal Clock Container, you added a Timed Event that occurred at 5 days. This Timed Event triggered two elements: one inside the Container and one outside the Container. In this model, the element inside the Container would be updated at 5 days. However, the element outside the Container would not be updated until 10 days (the first scheduled timestep after the event occurred). That is, the unscheduled update does not interrupt the clock outside the Container. The element outside the Container is still impacted by the event, but not until it is updated (e.g., due to a scheduled update, or due to an unscheduled update triggered outside of the Container).

It is also important to understand that while unscheduled updates within an Internal Clock Container do not cause elements outside the Container to update,
unscheduled updates *outside* the Container do cause elements inside the Container to update. That is, unscheduled updates within the Container cannot “flow” outside the Container, but unscheduled updates outside the Container do “flow” in.

While it is important to understand this behavior with regard to unscheduled timesteps automatically inserted by GoldSim, perhaps the most common use of an Internal Clock Container is to manually force a shorter timestep within the Container.

By default, the **Maximum time between updates** is set to 0 (and if you leave it unchanged or set it to a negative number, it will be ignored, as a non-positive length has no meaning).

When GoldSim starts to carry out the calculations for a Container that has an internal clock (or after GoldSim completes a timestep within the Container), and it is at a time we will call \( T_{\text{curr}} \), it must decide when to insert the next timestep inside the Container.

GoldSim uses the value in the **Maximum time between updates** field (referred to as \( DT_{\text{max}} \) below) to compute the next timestep inside the Container according to the following rules:

1. If \( DT_{\text{max}} \) is greater than \( DT_{\text{glob}} \) (the length of time until the next global timestep), the field is ignored and the next timestep is simply the regularly scheduled timestep \( (T_{\text{curr}} + DT_{\text{glob}}) \). (As a result, a Container with an internal clock can never have a timestep that is longer than the global timestep.)

2. If \( DT_{\text{max}} \) is less than \( DT_{\text{glob}} \), GoldSim computes an internal timestep length (an unscheduled update), \( DT_{\text{int}} \), equal to \( DT_{\text{glob}}/N \), where \( N \) is the smallest integer such that \( DT_{\text{glob}}/N \leq DT_{\text{max}} \). That is, GoldSim subdivides \( DT_{\text{glob}} \) such that the internal step is as large as possible, but less than or equal to \( DT_{\text{max}} \).

**Note:** Because certain events (e.g., triggered by a Timed Event or a Reservoir hitting an upper bound) are automatically represented by GoldSim with greater accuracy by allowing them to fall between scheduled timesteps, any events that occur could result in a smaller timestep than that computed using the rules presented above. That is, after GoldSim computes the next timestep based on the rules presented above, it then checks to see if an event would fall within that interval. If an event does fall into that interval, the timestep could be shortened in order to represent the event.

**Read more:** [Understanding Timestepping in GoldSim](page 455).

In most cases, you will specify the **Maximum time between updates** as a constant in order to specify a shorter timestep for a portion of your model (e.g., assigning a **Maximum time between updates** of 1 day to a Container in which the global timestep is 10 days). This variable can also be specified to vary dynamically in response to parameter values inside the Container.
Note: By default, time history displays only include scheduled updates, and will not capture unscheduled updates. However, GoldSim provides an option to do so (under a specified set of conditions) in the Advanced Time settings. This provides a mechanism to display and view the exact times at which unscheduled updates are inserted within a Container with an Internal Clock.

Read more: Including Unscheduled Updates in Time History Results (page 477).

Warning: When you assign an internal clock to a Container, the Treat as SubSystem feature is also automatically selected (and cannot be deselected unless you first turn off Internal Clock). That is, a Container with an internal clock, by definition, is treated as a SubSystem. Because a Container with an internal clock is treated as a SubSystem, this puts certain limitations on how these Containers can be used.

An example file which illustrates the use of Internal Clocks for Containers (InternalClock.gsm) can be found in the Containers subfolder of the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

Read more: Treating a Container as a SubSystem (page 135).

In some cases, events or other changes in the model may not fall exactly on a scheduled update. That is, some events or changes may actually occur between scheduled updates of the model. These trigger an “unscheduled update” of the model. Unscheduled updates are timesteps that are dynamically inserted by GoldSim during the simulation in order to more accurately simulate the system.

Read more: Understanding Timestepping in GoldSim (page 455).

Unlike scheduled updates, unscheduled updates do not normally appear in time history plots and tables. That is, although these timesteps may affect the results (e.g., by making them more accurate at the scheduled timesteps), unscheduled updates of the model are not saved and displayed. Only the scheduled updates are actually saved and displayed.

In some cases, however, it may be of interest to see the values of selected outputs at unscheduled updates. To facilitate this, GoldSim provides an option in the Advanced Time settings to save results at unscheduled updates (referred to as “high resolution” results). This option appears at the bottom of the Advanced Time Settings dialog (accessed from the Advanced… button on the Time tab of the Simulation Settings dialog):

By default, the checkbox labeled Include unscheduled updates (high resolution results) in scalar time histories is cleared. If you check this box, GoldSim will save unscheduled updates for time history results (referred to as “high resolution” results) under the following conditions:

- High resolution results can only be viewed from within Time History Result elements. If the output is not referenced by a Time History Result element, high resolution results will not be displayed.
Simulation Settings

– Chapter 7: Running a Model

Read more: Viewing Unscheduled Updates in Time History Result Elements (page 629).

– High resolution results are only available for single realization runs (Deterministic simulations or probabilistic simulation with a single realization).

– High resolution results are only available for scalar outputs.

Referencing Time in Edit Mode

When you place the cursor over an element, an input or an output prior to running a simulation, GoldSim displays a tool-tip showing the object's current (expected) value. If the object is a function of Etime, by default this calculation assumes a value of 0. You can modify the value used in such a calculation by editing Time to use for Edit Mode updates at the bottom of the Advanced Time Settings dialog (accessed from the Advanced… button on the Time tab of the Simulation Settings dialog):

Controlling When Weeks and Years Start

In some situations, you may wish to directly specify what GoldSim considers to be the first day of the week (First day of model week) and the first month of the year (Start of reporting year).

These can be specified at the bottom of the Advanced Time Settings dialog (accessed from the Advanced… button on the Time tab of the Simulation Settings dialog):

This information is then used in the following ways:

First day of model week

– When using “Weeks” as the Basic Step, and selecting “Calendar aligned” for the Alignment, the specified “First day of model week” controls where each new timestep is placed.

Read more: Specifying the Basic Step Length and Alignment (page 459).

– If you define a “weekly” Reporting Period, where the Periods are added is controlled by how you define the “First day of model week”. Weekly Periods are added at midnight on the specified first day of the week.

Read more: Defining Reporting Periods (page 461).

– The “DayOfWeek” Run Property (a number from 1 to 7) is controlled by how you define the “First day of model week”. The value 1 corresponds to the specified first day.

– The “WeekOfYear” and “YearOfWeek” Run Properties are used to track the week of the year. When these are incremented is a function of the “First day of model week”.

Referencing Time in Edit Mode

Controlling When Weeks and Years Start
Read more: Understanding and Referencing Run Properties (page 486).

Start of reporting year

- If you define an “annual” or “quarterly” Reporting Period, where the periods are added is controlled by how you define the “Start of reporting year”. Annual Periods are added on the first day of the month selected as the “Start of the reporting year”. Quarterly Periods are added on the first day of the month selected as the “Start of the reporting year”, and the first days of the months 3, 6 and 9 months after the selected month.

The Monte Carlo tab of the Simulation Settings dialog is used to specify the Monte Carlo options for a simulation. This consists of those parameters which control how probabilistic simulations will be carried out, such as the number of realizations, and how the sampling of Stochastic variables is to be carried out (e.g., whether Latin Hypercube sampling is to be used).

Note: Appendix A provides an introduction to probabilistic simulation techniques, and introduces the basic terminology required to understand the Monte Carlo options presented in this dialog. The discussion that follows assumes that you are familiar with the concepts discussed in that appendix.

There are two basic types of simulations that you can carry out with GoldSim:

- In a Probabilistic Simulation, one or more input parameters are represented as probability distributions (and/or some processes are represented as being stochastic) and a simulation is carried out by running the model multiple times.

- In a Deterministic Simulation, single values are used for each input parameter, and a simulation is carried out by running the model a single time.

These two types of simulation are discussed in detail in the sections below.

To run a probabilistic simulation in GoldSim, select the Probabilistic Simulation radio button in the Monte Carlo tab of the Simulation Settings dialog (which is the default):
You must then select the number of realizations (# Realizations), which defaults to 1.

If you check the Run the following Realization only checkbox, GoldSim runs only a single realization (specified in the Realization field immediately to the right). In this case, the random number sequence that is used is based on the Realization specified and (if Latin Hypercube Sampling is on) the total number of realizations (# Realizations). Hence, if you first run a 100 realization simulation, and then run a second simulation with Run the following Realization only checked (and, for example, Realization set to 14), the result of the second simulation will be identical to the 14th realization of the first simulation.

This is used in cases where you first run a large number of realizations (but only save a limited number of time histories) and then want to examine the details (i.e., the time history) for a particular realization for which you did not save time histories, but which exhibited some interesting behavior.

GoldSim provides two primary ways in which you can control the manner in which the Monte Carlo sampling of Stochastic elements takes place:

- implementing Latin Hypercube sampling; and
- repeating sampling sequences.

If Use Latin Hypercube Sampling is checked (the default), GoldSim uses Latin Hypercube sampling. This sampling method causes probability distributions (i.e., Stochastic elements) to be divided into a number of equally likely “strata”. This has the effect of ensuring that the probability space of the parameter is uniformly spanned. When using Latin Hypercube sampling, you can choose whether each stratum is sampled randomly (“Use random points in strata”), or if the mid-point of each stratum is used (“Use mid-points of strata”).
In addition to Latin hypercube sampling, you can also specify importance sampling for individual Stochastic elements. Importance sampling allows you to over-sample the tails of a distribution.

Read more: Applying Importance Sampling to a Stochastic Element (page 181).

Note: Importance sampling changes the weight of realizations so they are not all equal. If using importance sampling and specifying realization weights, the relative realization weights are applied after the importance sampling is applied.

If Repeat Sampling Sequences is checked (the default), a repeatable random number sequence is used (based on the specified Random Seed), allowing you to repeat the simulation exactly. If this box is cleared, the random number is based on the current date and time, such that the random number sequence will be different every time you run the model.

Note: Each element that exhibits random behavior (e.g., Stochastics, Delays) is assigned an “internal” random number seed when it is first created and this is stored with the element. The random number used for that element is then a function of both its “internal” random number seed, the number specified in the Random Seed field, and the realization number. Hence, if you create a file and then copy it to another person, when they run it, they will produce the same random numbers as you (as long as you both have the same number specified for the Random Seed). However, if that person builds the exact same model independently, you will always produce different random numbers, since the “internal” seeds for the Stochastics will be different.

The Specify Realization Weights option is an advanced feature that provides the ability to manually specify the relative weight for each realization. Realization weights are used to compute statistics and create the CDF. If doing simple Monte Carlo sampling, each realization, by definition, has an equal weight (i.e., each realization is equally likely). The primary reason to manually specify realization weights is to carry out customized importance sampling of Stochastic elements.

Read more: Customized Importance Sampling Using User-Defined Realization Weights (page 1013).

If you select the Specify Realization Weights box, an Insert link dialog appears, and you must select an element that defines the relative realization weights (e.g., equally weighted realizations could all have weight of 1). It must be a non-negative, dimensionless scalar, and GoldSim will use the value at the end of the realization. Obviously, to be of any value, this must vary from realization to realization. If you simply link to a value that is independent of the realization (e.g., a constant), it will have no impact on the weights.

The manner in which GoldSim implements random seeds, as well as other details of the sampling algorithm (such as Latin Hypercube sampling and importance sampling) are discussed in Appendix B.

The Result Options… button provides access to a dialog for controlling how various Monte Carlo results are saved and displayed.

Read more: Controlling Monte Carlo Result Options (page 482).
The Monte Carlo Result Display Properties dialog allows you to control how Monte Carlo results (simulations with multiple realizations) are saved and displayed:

This dialog can be accessed from the Monte Carlo tab of the Simulation Settings dialog (via the Result Options... button), and is also accessible from result properties dialogs.

**Read more:** Viewing and Editing Result Properties (page 564).

The top section of the dialog (History Statistics) is used to specify how time history results with multiple realizations will be displayed. In particular, it allows you to define the percentiles of a particular presentation option for Monte Carlo time histories (referred to as probability histories), and also allows you to define a default statistic (Default Display Statistic) that is the default when viewing statistics for multiple realization simulations.

**Read more:** Viewing Probability Histories for Multiple Realizations (page 600); Viewing Time Histories of Multiple Realizations (page 597).

The bottom portion of the dialog (Realization Classification and Screening) allows you to define “categories” of realizations (e.g., “realizations in which X > 10”). You can then choose to show these categories on charts in different colors and/or choose to screen certain categories entirely from result displays.

**Read more:** Classifying and Screening Realizations (page 577).

To run a deterministic simulation in GoldSim, select the Deterministic Simulation radio button in the Monte Carlo tab of the Simulation Settings dialog:
In a deterministic simulation, a single realization will be carried out using a deterministic value for each Stochastic element. When you do so, it is necessary to specify what deterministic values are to be used for each Stochastic element.

As can be seen, there are three options for running a deterministic simulation:

**Element Deterministic Values**: In this case, the deterministic value explicitly specified in the Stochastic element property dialog is used for each Stochastic element.

**Element Mean Values**: In this case, the mean value of each Stochastic is used (overriding any deterministic value specified for each Stochastic).

**Specified Quantile**: In this case, the specified quantile (a number between 0 and 1) of each Stochastic is used (overriding any deterministic value specified for each Stochastic).

*Read more:* [Specifying a Deterministic Value for a Stochastic](page 180).

**Note**: If you are resampling a Stochastic during a Deterministic Simulation, its value will not change (i.e., it will not behave stochastically), unless the distribution itself is changed as a function of time.

Stochastic elements are not the only elements that behave probabilistically. Several other elements in GoldSim also do so. In a Deterministic Simulation, these elements also behave deterministically (i.e., their behavior is no longer random):

- Randomly time-shifted Time Series are sampled from the middle of the historic data (i.e., the starting point is based on the median value of the series).
Defining and Referencing Global Properties

In some cases, you may want to define global properties that can be referenced throughout your model. Of course, one way to do this is to simply define Data elements at the global level of the model (i.e., outside of any localized Containers).

GoldSim also provides a more convenient way to do this via the Globals tab of the Simulation Settings dialog:

This dialog allows you to add Global variables (referred to as Global properties) to your model. This is done by pressing the Add... button. When you do so, the following dialog is displayed:
The **Name (ID)** is the name by which the Global variable can be referenced throughout your model. This Name has the same restrictions as an element ID. The **Description** is an optional description of the variable. You must then specify the **Display Units** and **Type** information (e.g., value/condition, scalar/vector/matrix) for the variable. Finally, you specify the **Definition** for the variable. Typically, this will be a single value. However, it can also be an equation (involving constant values). This field, cannot, however, reference any other outputs in the model.

Once you have added a Global property, you can subsequently reference the variable by using the specified **Name** that you defined previously, preceded by a ~. For example, if you create a Global called X, you could reference it in input fields anywhere within the model as ~X.

Global properties are special instances of what are referred to as locally available properties in GoldSim. Locally available properties are accessed by using a ~. Global properties are available in the Insert Link dialog within the Available Properties folder:

![Select the output to link to](image)

*Read more:* [Understanding Locally Available Properties](#) (page 808).

---

**Note:** Because a Global property is always referenced using a ~, a Global property can share a name with an element. For example, if a Global property existed named X and a Data element also existed with the same name, there would be no conflict. The Global property would be referenced as ~X, and the Data element's output would be referenced as X.

**Note:** The Global properties tab is also used to "hold" the input interface for a SubModel that has been exported as a standalone model.

*Read more:* [Using SubModels to Embed Models Within Models](#) (page 973).

The **Information** tab of the Simulation Settings dialog displays some summary information regarding your simulation:

---

**Viewing and Editing Model Summary Information**
The top part of the dialog provides two fields (Author and Analysis description) which allow you to identify the author and provide a brief description for your run. These fields have two uses:

- They are written to the Run Log; and
- They can be inserted into result charts using keywords.

Read more: The Run Log (page 548); Using Keywords in Styles (page 717).

The bottom part of the dialog provides some useful summary statistics for the model, including the number of model elements and graphical components, and the degree of subcontainment. The number of levels of subcontainment does not refer to the number of Containers; it refers to the number of hierarchical levels of Containers. For example, if the model contained Containers B and C (in parallel), the model would have 1 level of subcontainment; if B contained another Container named C, the model would have 2 levels of subcontainment.

The dialog also indicates the total size of the results being saved for all elements within the model. This information can be very useful in helping you to manage the size of the model file.

Understanding and Referencing Run Properties

Within a simulation model, you often need to reference within input expressions internal model variables like the elapsed time, the date/time, and the realization number. To facilitate this, GoldSim provides a number of special reserved names, referred to as Run Properties, that can be directly referenced in expressions. The most commonly used Run Properties are ETime (the simulated elapsed time) and DateTime (the simulated calendar date-time).
Read more: Referencing Time in GoldSim (page 101).

In addition to ETime and DateTime, GoldSim also provides many additional Run Properties that can be referenced in your model.

You can enter the names of the Run Properties directly when creating expressions (just as you would enter the name of an output), or you can insert them into an expression using the Insert Link dialog (accessed by right-clicking in an input field and selecting Insert Link... from the menu).

As can be seen above, at the top of the dialog is a folder labeled “Run Properties”. If you expand this folder, you will see that all of the Run Properties are organized into four categories (Calendar Time, Elapsed Time, Simulation and Reporting Periods).

Expanding any particular category folder lists all of the Run Properties in that category:

As can be seen, the icon for the various Run Properties is different than that used for element outputs. This is because the Run Properties don’t actually represent the output of any particular element. They are special model properties that are “broadcast” throughout the model and can be referenced anywhere.

Note: The color of the symbol changes if the Run Property is actually being referenced in a model. If it is not being referenced, it is gray. Otherwise, it is blue. In the example above, the Run Property “Month” is being referenced.
Note: Run Properties are an instance of a class of variables called Locally Available Properties. Locally Available Properties derive their name from the fact that in specialized cases, they may only be available, or may take on different values (i.e., be over-ridden), in “local” parts of your model (e.g., within particular Containers). In the case of Run Properties, they are always available throughout the model, but some of these properties are overridden locally if a Container is defined to have an internal clock (a local timestep).

Read more: Understanding Locally Available Properties (page 808); Specifying Containers with Internal Clocks (page 474).

Note: All the Run Properties are protected names, so, for example, you cannot create an element named “ETime” or “DateTime”.

Note: You cannot directly save a time history of a Run Property, or reference it in a Result element. If you wish to do so, link it to an Expression and save or reference the Expression element in the Result element.

The four categories of Run Properties are discussed in the sections below.

GoldSim provides a number of special reserved names, referred to as Run Properties, that can be directly referenced in expressions. Some Run Properties are associated with Calendar Time. That is, they return information regarding the simulated date and time of a simulation.

These can be found under the “Calendar Time” folder:

Note: Because even Elapsed Time simulations have a specified Start Time, Calendar Time Run Properties are available for both Elapsed Time and Calendar Time simulations.
The Calendar Time Run Properties are described below:

**DateTime**: This represents the elapsed time from a reference point (December 30, 1899 at 00:00:00 – that is, one day before the beginning of the twentieth century). It has units of time.

**Note**: You can compare the DateTime Run Property to an actual calendar date by enclosing the date in quotation marks within an expression. For example, this if statement would compare the simulated date to the specified date of 15 April 2012: `if(DateTime > “15 April 2012”, 1, 2)`.

**Year**: This is the calendar year. It is unitless.

**Quarter**: This is the calendar quarter (an integer between 1 and 4), assuming the first quarter starts on January 1. It is unitless.

**Month**: This is the calendar month (an integer between 1 and 12). It is unitless.

**WeekOfYear**: This is the week number for the current YearOfWeek (see below). It is a unitless integer between 1 and 53. Each week starts (and hence WeekOfYear changes) on the “first day of model week” specified in the Advanced Time Settings dialog.

**YearOfWeek**: This is the nominal year for the current WeekOfYear (see above). It is a unitless integer. YearOfWeek is incremented at the start of the first week that has the majority of its days in the next calendar year. Thus at the beginning of the first week that starts after December 28, YearOfWeek is incremented to the next calendar year, and WeekOfYear is reset to 1.

**DayOfYear**: This is the calendar day of the year (an integer number between 1 and 366). It is unitless.

**DayOfMonth**: This is the calendar day of the month (an integer between 1 and 31). It is unitless.

**DayOfWeek**: This is the calendar day (an integer between 1 and 7). It is unitless. The weekday corresponding to the value 1 is determined by the specified “First day of model week” specified in the Advanced Settings of the Time tab of the Simulation Settings dialog.

**Hour**: This is the calendar hour (an integer between 0 and 23). It is unitless.

**Minute**: This is the calendar minute (an integer between 0 and 59). It is unitless.

**Second**: This is the calendar second (an integer between 0 and 59). It is unitless.

**DaysInMonth**: This is the number of days in the current calendar month (an integer between 28 and 31). It is unitless.

**DaysInYear**: This is the number of days in the current calendar year (either 365 or 366). It is unitless.

**StartTime**: This is the simulated Start Time for the simulation (expressed relative to December 30, 1899 at 00:00:00). It has units of time and is
generally viewed as a date. It can be compared to other dates in expressions.

**Note:** An example file which illustrates the use of Calender Time Run Properties (RunProperties.gsm) can be found in the General Examples/Running folder in your GoldSim directory (accessed by selecting **File** | **Open Example**... from the main menu).

**Note:** The Run Properties Year, Quarter, Month, WeekofYear, DayOfMonth, DayofWeek, DayofYear, Hour, Minute and Second can be particularly useful for triggering events that must occur exactly every year, quarter, month, week, day, hour, minute or second. For example, if you had a 1 day timestep (or smaller), defining an OnChanged trigger as DayOfYear or DayOfMonth would force the element to be triggered every day. An example file which illustrates this (RunProperties.gsm) can be found in the General Examples/Running folder in your GoldSim directory (accessed by selecting **File** | **Open Example**... from the main menu).

**Read more:** [Specifying Triggering Events](#) (page 356).

GoldSim provides a number of special reserved names, referred to as Run Properties, that can be directly referenced in expressions. Some Run Properties are associated with Elapsed Time. That is, they return information regarding the simulated elapsed time.

These can be found under the “Elapsed Time” folder:

The Elapsed Time Run Properties are described below:

- **Etime:** This is the elapsed time since the start of the simulation. It has units of time.

- **RTime:** This is the remaining time in the simulation (computed as total duration minus the elapsed time). It has units of time.
**EYear**: This is the integer number of *average* years that have elapsed since the start of the simulation (hence, at the start of the simulation, it is zero). For the purposes of the calculation, an average year is computed as 365.25 days. It is unitless.

**EQuarter**: This is the integer number of *average* quarters that have elapsed since the start of the simulation (hence, at the start of the simulation, it is zero). For the purposes of the calculation, an average quarter is assumed to have 365.25/4 = 91.3125 days. It is unitless.

**EMonth**: This is the integer number of *average* months that have elapsed since the start of the simulation (hence, at the start of the simulation, it is zero). For the purposes of the calculation, an average month is assumed to have 365.25/12 = 30.4375 days. It is unitless.

**EDay**: This is the integer number of days that have elapsed since the start of the simulation (hence, at the start of the simulation, it is zero). It is unitless.

**EHour**: This is the integer number of hours that have elapsed since the start of the simulation (hence, at the start of the simulation, it is zero). It is unitless.

**EMinute**: This is the integer number of minutes that have elapsed since the start of the simulation (hence, at the start of the simulation, it is zero). It is unitless.

**ESecond**: This is the integer number of seconds that have elapsed since the start of the simulation (hence, at the start of the simulation, it is zero). It is unitless.

---

**Note**: An example file which illustrates the use of Elapsed Time Run Properties (RunProperties.gsm) can be found in the General Examples/Running folder in your GoldSim directory (accessed by selecting **File** | **Open Example...** from the main menu).

---

**Run Properties: Simulation**

GoldSim provides a number of special reserved names, referred to as Run Properties, that can be directly referenced in expressions. Some Run Properties return basic information about the simulation.

These can be found under the “Simulation” folder:
The Simulation Run Properties are described below:

**TimestepLength**: This is the length of the current timestep. More specifically, it is the time since the last model update. (At time = 0, the TimestepLength is set to the length of the first scheduled timestep). Note that under some circumstances (e.g. when creating “unscheduled updates” to better represent discrete events, or to dynamically adjust the timestep during a simulation), a model can be updated between scheduled timesteps, and this will be reflected by the TimestepLength Run Property.

**Read more**: Understanding Timestepping in GoldSim (page 455); Dynamically Controlling the Timestep (page 471).

**Note**: At Etime = 0, TimestepLength returns the anticipated length of the first step (i.e., the length of the first scheduled timestep).

**Note**: An example file which illustrates the use of TimestepLength (RunProperties.gsm) can be found in the General Examples/ Running folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

**Realization**: This is the realization number. For a deterministic simulation, it is always equal to 0. It is unitless.

**NumOfReal**: This is the total number of realizations being carried out in the simulation.
Note: NumOfReal actually represents the value entered into the # Realizations field in the Simulation Settings dialog, and this is not necessarily the same as the number of realizations that are carried out. In some cases (e.g., if you specify a Deterministic run, if you instruct GoldSim to run a particular realization, or if you are doing distributed processing), GoldSim will actually only run a single realization, but would still report the value from the # Realizations field as the NumOfReal.

**Run Properties: Reporting Periods**

**SimDuration:** This is the total duration (i.e., the amount of simulated time) represented by the simulation.

GoldSim provides a number of special reserved names, referred to as Run Properties, that can be directly referenced in expressions. Some Run Properties are associated with Reporting Periods that you have defined in the Simulation Settings dialog.

*Read more:* [Defining Reporting Periods](page 461).

These can be found under the “Reporting Periods” folder:

![Screenshot of GoldSim run properties](image)

The Reporting Periods Run Properties are described below:

- **ReportingMonth:** This is the current reporting month, relative to the month specified as the “Start of reporting year” in the the Advanced Settings of the Time tab of the Simulation Settings dialog. For example, if the “Start of reporting year” was defined as April, and the current month in the simulation was June, ReportingMonth would be 3; if the current month in the simulation was April, ReportingMonth would be 1. This Run Property is always available (even if no Reporting Periods are defined). It is unitless.

*Read more:* [Controlling When Weeks and Years Start](page 478).

- **ETime_Major** (where **Major** is the user-defined Major Period Label): This is elapsed time since the start of the current Major Reporting Period. Note that at the start of a new Major Period, this displays the duration of the just-completed period. It has units of time, and is only available if a Major Reporting Period is defined.
Saving Outputs as Results

A GoldSim simulation has the potential to produce a large amount of results. GoldSim does not produce separate input and output files. The model definition and the results generated by the model are all stored in one file: the gsm (GoldSim Model) file. Hence, for large models, particularly those with many timesteps and/or multiple realizations, when in Result Mode, your model file could potentially become very large.

In most cases, however, you will only be interested in a small subset of these results, and GoldSim provides powerful tools to allow you to selective about which results you save, and when they are to be saved.

Before you run a model, you select the results that you wish to save. With regard to saving results, there are two things that you must specify:

- **Which** specific results do you want to save (i.e., which outputs of which elements)?
- **When** do you want to save these results?

Specifying Results to be Saved

**Note:** An example file which illustrates the use of Reporting Period Run Properties (RunProperties.gsm) can be found in the General Examples/Running folder in your GoldSim directory (accessed by selecting **File | Open Example...** from the main menu).
Within GoldSim, there are two fundamental types of results:

**Final Value Results**: These are usually used to view results involving multiple realizations (or items in an array), and allow you to display distributions, multi-variate results (e.g., scatter plots) and array plots. By default, “Final Value” type results are only available at the final time point in the simulation. In some cases, however, you may also want to view distributions or scatter plots at other points in time (other than the end of the simulation). GoldSim facilitates this by allowing you to create Capture Points at which these results are also made available.

*Read more: Creating Capture Points for Final Value Results* (page 469).

**Time History Results**: These are saved at selected timesteps, and allow you to display how an output changes as a function of time.

All element property dialogs have check boxes (at the bottom of the dialog) to specify whether output(s) of the element are to be saved. You can save the **Final Values** (the values at the end of each realization in the simulation) and/or **Time History** (the value at selected timesteps throughout the simulation):

In general, by default when you create a new scalar element, the Save Results checkboxes will be checked on. For array elements, the **Time History** checkbox defaults off.

---

**Note**: For some GoldSim extension modules, the default for some elements is not to save Time History (or Final Value) results.

The **Final Values** checkbox always controls whether Final Value results are saved. However, the **Time History** checkbox can be overridden. In particular, it is always applied for single realization runs, but is overridden for multiple realization runs. **Time History results for multiple realization runs are only saved for outputs that are connected to Result elements.**

*Read more: Creating and Using Result Elements* (page 570); *Viewing Time Histories of Multiple Realizations* (page 597).

That is, regardless of the setting of the **Time History** checkbox, if you are running multiple realizations, the Time History box is ignored, and unless an output of the element is connected to a Time History Result element, no time history results will be saved at all for the element.

In fact, in such a case, the Save Results section would look like this:

In this example, multiple realizations are being run, and none of the outputs are referenced by a Time History Result element.

In this particular case, because the simulation involves multiple realizations, the “Time History” checkbox is replaced by a “Monte Carlo Histories” checkbox. This checkbox is grayed out, since it is only used for information purposes. You cannot edit it directly, and its appearance is completely determined by whether or not the element’s outputs are connected to a Time History Result element.
If no outputs of the element are connected to a Time History Result element, the box is cleared (as above). If all the outputs of the element are connected to a Time History Result element, the box will be checked:

In this example, multiple realizations are being run, and all of the element’s outputs are referenced by a Time History Result element.

Some elements have multiple outputs. For these elements, you can control which outputs you wish to save. A check indicates that all of the outputs of the element are being saved.

If you wish to save some of the element’s results (but not all), and you are not running multiple realizations, you can use the context menu of the output interface (or the browser) to turn one or more of the outputs on or off:

In this example, multiple realizations are being run, and some, but not all, of the outputs are referenced by a Time History Result element.

Similarly, if you wish to save some of the element’s results (but not all), and you are running multiple realizations, you can do so by connecting some (but not all) of the outputs to Time History Result elements:

In this example, multiple realizations are being run, and some, but not all, of the outputs are referenced by a Time History Result element.

You can globally deactivate all of the result flags for the elements inside a Container (that are not connected to active Result elements) from the Container dialog.

Read more: Controlling Result Flags for Elements in the Container (page 141).

The Information tab on the properties dialog for a Container displays the total amount of disk space required to store the results for the elements within the Container. The total amount of disk space required to save the results for the entire model is displayed in the Information tab of the Simulation Settings dialog.
Running and Viewing the Status of a Simulation

GoldSim User’s Guide

Chapter 7: Running a Model • 497

Highlighting Outputs that Will be Saved

**Note:** When Time History results for specific outputs are saved (either because you are running a single realization and have checked the Time History box, or because you have referenced an output in a Time History Result element), by default GoldSim saves the values of outputs at the beginning and end of the simulation, and at every Basic Step (and Reporting Period). You can instruct GoldSim to only save results at selected timesteps (e.g., every tenth Basic Step) when defining timestepping options in the Simulation Settings dialog.

**Read more:** Understanding Timestepping in GoldSim (page 455); Specifying When Time History Results Will Be Saved (page 464).

In some cases, you may want to quickly ascertain which outputs in a model are set to be saved. This is important, for example, if you discovered (from the Container dialog) that a large amount of disk space will be required to store all of the results, and you wish to turn some of the save flags off.

GoldSim provides a quick way to do this. In particular, you can instruct GoldSim to highlight all elements (and outputs) for which results are to be saved. You do this by selecting **View | Highlight Saved Results** from the main menu. You can toggle highlighting on and off separately for Final Values and Time Histories. After doing so, elements and outputs for which results will be saved will be displayed in **bold** text within the browser. Moreover, the output ports for elements for which results will be saved will be changed to green.

**Note:** By default, only elements are displayed in the browser. If you also wish to view all of the elements’ inputs and outputs in the browser, right-click anywhere in the browser window and select **Show Element Subitems**. You can also do this directly from the main menu by selecting **View | Show Element Subitems**.

Archiving a File with Results

GoldSim provides a utility that allows you to **archive** a file. Archiving a file saves a copy of it under a different filename (which you provide) while keeping the current file open. This can be useful if you want to save the current file (with results), and then edit the model and rerun it. You archive a file by selecting **File | Save Copy As**... from the main menu.

Running and Viewing the Status of a Simulation

GoldSim simulations are run and controlled using the **Run Control toolbar**:

(You can also run and control some aspects of a simulation from the main menu and/or by using shortcut keys).

In addition to running (and resetting) a model, the Run Control toolbar can also be used to pause and resume a simulation, and/or step through the simulation one timestep or one realization at a time.

The status of a run can be monitored in the status bar:
While the model is running, the status bar displays the status of the run.

The topics below discuss the details of how you can run, control and monitor the status of your simulations.

In order to fully understand how to run, control and monitor the status of your simulations, it is necessary to first understand the various simulation modes (model states) that a GoldSim model can be in at any given time.

At any given time, a GoldSim model is in one of four states (referred to as modes):

- **Edit Mode**: The model is currently being edited and does not contain any simulation results.
- **Run Mode**: The model is currently running. In this mode, you can pause the simulation and view the model, but elements cannot be added, deleted or edited. Only cosmetic changes can be made to the model (e.g., adding text, moving elements around the screen).
- **Result Mode**: The model has been run and contains simulation results. In this mode, you can navigate the model and view results, but no elements (other than Result elements) can be added, deleted or edited. Only cosmetic changes can be made to the model (e.g., adding text, moving elements around the screen).
- **Scenario Mode**: This is a special mode that only exists if you are using GoldSim’s scenario feature. To be in this mode, the model must contain at least one scenario that has been defined by the user, and at least one scenario must have scenario results. In this mode, you can navigate the model and compare scenario results, but no elements (other than Result elements and Scenario Data elements) can be added, deleted or edited. Read more: [Creating, Running and Comparing Scenarios](page 504).

The mode that the model is in is clearly identified in the status bar (in the lower left-hand corner of the GoldSim window). In addition, the status bar takes on a different color in each mode:

- **Edit Mode**: The model is currently being edited and does not contain any simulation results. The status bar is blue.
- **Run Mode**: The model is currently running. In this mode, you can pause the simulation and view the model, but elements cannot be added, deleted or edited. Only cosmetic changes can be made to the model (e.g., adding text, moving elements around the screen). The status bar is pink.
- **Paused**: The model is in Run Mode, but it is paused. The status bar is rust brown.
- **Result Mode**: The model has been run and contains simulation results. In this mode, you can navigate the model and view results, but no elements (other than Result elements) can be added, deleted or edited. Only cosmetic changes can be made to the model (e.g., adding text, moving elements around the screen). The status bar is green.
- **Scenario Mode**: This is a special mode that only exists if you are using GoldSim’s scenario feature. To be in this mode, the model must contain at least one scenario that has been defined by the user, and at least one scenario must have scenario results. In this mode, you can navigate the model and compare scenario results, but no elements (other than Result elements and Scenario Data elements) can be added, deleted or edited. The status bar is orange.

One of the key ways that the various modes can be distinguished is by whether or not you can edit your model while in that mode. As pointed out above, elements can only be edited while in Edit Mode (and under special...
Running and Viewing the Status of a Simulation

Carrying Out a Simulation (Run Mode)

circumstances, in Scenario Mode). In the other two modes, element editing is completely disabled.

Run Mode and Result Mode are discussed in more detail in the following sections.

Once you have finished editing your model, you can run the model by selecting Run | Run Model from the main menu, pressing F5, or pressing the Run button in the Run Control toolbar. This causes GoldSim to check the model for errors (to see if it can be run).

Run button

If an error is found in the model such that it cannot be run, GoldSim will display a fatal error message describing the problem. The error message will identify the element that is causing the problem:

Pressing the Edit… button from this dialog returns the model to Edit Mode and immediately opens the properties dialog of the element which prevented the model from running.

If multiple errors are discovered, the error message dialog will look like this:

Pressing the Edit… button from this dialog returns the model to Edit Mode and immediately opens the properties dialog of the element associated with the selected error. Selecting Show Full Path shows the full path to the input causing the error.

If GoldSim does not detect any errors, the model will be placed into Run Mode.

When GoldSim is in Run Mode, it can have one of three states (the state or status is displayed on the left side of the status bar):

- Ready
- Running
Run Mode States

When GoldSim is in Run Mode, it can have one of three states (the state or status is displayed on the left side of the status bar).

- **Paused**
  - Paused
  
  In the “Ready” state, all elements have been checked, and the simulation is ready to begin.

  By default, when you press the Run button (or F5), the model skips through the “Ready” state and immediately starts the simulation. However, you can instruct GoldSim to pause and wait in the “Ready” state by holding down the Ctrl key when you press the Run button (or F5). You must then press F5 again to start the simulation. This is only of value if you wish to pause and step through a simulation.

  Read more: Pausing and Stepping through a Simulation (page 502).

  In the “Ready” state, the toolbar is still blue (same color as if in Edit Mode), but simulation status information is now displayed (all set to initial values):

  ![Status Bar](image)

  If you want to return to Edit Mode when you are in the “Ready” state (to edit the model), press the Edit button on the Run Control toolbar (the first button from the left in the toolbar), or press F4.

  Note: A model can not be opened in Run Mode. It can only be opened in Edit Mode, Result Mode or Scenario Mode. Hence, if you save and close a model while it is in the “Ready” state (Run Mode), the next time it opens it will be placed in Edit Mode.

- **Running**
  - Running
  
  Once the simulation is started, the status bar changes color and “Running” is displayed on the left side of the bar. Next to “Running”, an estimate of the progress of the simulation (in terms of what fraction is currently complete) is displayed. The progress is displayed both as a percentage, and as a progress bar (on the right side of the status bar):

  ![Status Bar](image)

  Note: For a single realization run, the progress represents the fraction of scheduled timesteps that have been completed. For a multi-realization run, the progress represents the fraction of realizations that have been completed (accounting for the fraction of the current realization completed). Progress is an estimate because unscheduled timesteps could potentially occur that cause some portions of a realization (or different realizations) to require significantly more time to complete. (Interrupt elements could also shorten some realizations in an unpredictable way).

  Read more: Understanding Timestepping in GoldSim (page 455).
More detailed information on the simulation status is also displayed. In particular, the status bar displays the progress of the simulation in terms of the actual elapsed time (i.e., how long has the simulation been running), the simulated time for the current realization, and the current realization being run:

The Run Control toolbar changes its appearance when the model starts running. Prior to the start of the simulation, it looks like this:

Once the simulation begins, the toolbar looks like this:

The Abort (first button from the left) and Pause button (second button from the left) can be used to abort and pause the simulation, respectively.

Just to the right of the Run button are two additional buttons for stepping through the simulation one timestep or one realization at a time. To the right of these is a slider that can be used to artificially slow down the speed of a simulation.

Read more: Aborting a Simulation (page 503); Pausing and Stepping through a Simulation (page 502).

Paused

If you press the Pause button while a simulation is running, the status will change to “Paused”, and the status bar will indicate this:

The Run Control toolbar changes its appearance:

The Run button can be used to restart the run. Just to the right of the Run button are two additional buttons for stepping through the simulation one timestep or one realization at a time.

Read more: Pausing and Stepping through a Simulation (page 502).

Note that if a fatal error occurs while your simulation is running (e.g., if you set up a model so that an element tries to divide by zero), a dialog explaining the error is shown:
Running and Viewing the Status of a Simulation

Pausing and Stepping through a Simulation

The model will actually be placed in Result Mode (as any realizations that completed prior to the error can be viewed).

Once a simulation begins, the toolbar looks like this:

![Run Control Toolbar]

The Pause button (second button from the left) can be used to pause the simulation.

If you press the Pause button in the Run Control toolbar while a simulation is running, the model will be paused, and this will be indicated in the status bar:

![Status Bar]

Note: GoldSim also provides the option to pause the model immediately (in the “Ready” state, before the simulation begins) by pressing the Ctrl key while pressing the Run button or F5.

Once a model is paused, you can browse the model and view results in the same manner you would if you were in Result Mode.

Read more: Viewing Results (Result Mode) (page 504).

When a model is paused, tool-tips displayed when you hold the cursor over an element or output show the Last Value calculated prior to pausing the model. Since a model can be paused in the middle of a timestep (such that some of the elements have been updated for the timestep and others have not), the tool-tip indicates whether or not the value has been updated for the current timestep.

If you are viewing a Time History Result element while running the simulation, GoldSim will plot and update the time history of the current realization whenever you pause the model.

Read more: Viewing a Result Element (page 573).

While a model is paused, you can navigate it, open property dialogs, and view result plots. You cannot, however, edit the model in any way that would change its behavior (e.g., you cannot change values in property dialogs or add or delete elements). You also cannot close or save a model while it is paused.

Note: You can change values for some elements (Data elements) while a model is paused if and only if the element is linked to a Dashboard control. Dashboard controls are discussed in the GoldSim Dashboard Authoring Module User's Guide.

While the model is paused, the Run Control toolbar changes its appearance:

![Run Control Toolbar (Paused)]

Pressing the Abort button (first button from the left) aborts the simulation.

Read more: Aborting a Simulation (page 503).

The Run button can be used to restart the run. Just to the right of the Run button are two additional buttons for stepping through the simulation one timestep or one realization at a time:
Running and Viewing the Status of a Simulation

If you press the button for simulating one timestep, GoldSim will complete the next timestep and then pause. If you press the button for simulating one realization, GoldSim will complete the next realization and then pause.

On the right side of the Run Control toolbar is a slider that can be used to artificially slow down a simulation. This is useful primarily if you want to pause the simulation at a specific point, or you have set up some results plots (or Dashboards) that you want to be able to view evolving slowly over the course of a simulation.

**Warning:** By default, the slider is all the way to the right (Fast). If you want the simulation to run as quickly as possible, make sure that the slider stays in this position. If you move it to the left, the simulation will take longer!

**Aborting a Simulation**

Once a simulation begins, the toolbar looks like this:

If the simulation is paused, the toolbar looks like this:

The **Abort** (first button from the left) can be used to abort the simulation.

If you press the **Abort** button while a model is running (or paused), the following dialog will be displayed:

The dialog provides you with two options:

**Keep Results:** This button will keep the existing results and place the model in Result Mode.

**Discard Results.** This button will discard all results, and place the model in Edit Mode.

If you are aborting during the first realization and you choose to keep results, in order to generate results for the remainder of the partial realization, GoldSim will skip to the end of the realization (i.e., skip all further calculations for the realization, keeping the values of all elements constant at their last compute value).

If you are running multiple realizations, and abort after the first realization, you will have the choice to **Include results of current partially completed realization.** If this box is cleared (the default) and you choose to keep results, the current partially completed realization will be discarded (only the completed realizations will be saved). If this box is checked, GoldSim will generate results for the remainder of the partial realization by skipping to the end of the realization (i.e., skipping all further calculations for the
realization, keeping the values of all elements constant at their last compute value).

**Warning:** When you abort a simulation and include the results from the last (partially complete) realization, you should be careful as to how you interpret the results. In particular, because the last realization was not complete, it may be inappropriate to combine it with or compare it to other (full) realizations.

### Viewing Results (Result Mode)

After a simulation is complete, the model will be placed in Result Mode. The status bar turns green and “Results” is displayed on the left side:

![Result Mode Status Bar](image)

More importantly, editing the model will be disabled.

In addition, the cursor takes on a different appearance (it turns green).

When in Result Mode, elements with outputs that have been saved are identified in two ways: 1) the elements (and their outputs) are **bold** in the browser (and output interfaces); and 2) the element output ports in the graphics pane are **green**.

When a model is in Result Mode, you can navigate it, open property dialogs, and view result plots. You can also move elements around within the graphics pane, add graphics (e.g., text, images), and add and edit result elements. You cannot, however, edit the model in any way that would change its behavior (e.g., you cannot change values in property dialogs or add or delete elements).

**Read more:** [Chapter 8: Displaying Results in GoldSim](page 555).

In Result Mode, the Run Control toolbar looks like this:

![Run Control Toolbar](image)

If you press **F4**, choose **Run|Return to Edit Mode** from the file or press the first button from the left in the Run Control toolbar, the results will be discarded and the model will be returned to Edit Mode.

### Creating, Running and Comparing Scenarios

GoldSim provides a specialized capability that allows you to create, run and compare different scenarios for your model. Scenarios are differentiated by having different sets of input data. In particular, different scenarios have different values for one or more Data elements. GoldSim’s scenario modeling capability allows you to directly compare results generated by your model by different sets of input parameters. In effect, when you use this capability, your model can store (and subsequently compare) multiple sets of inputs and outputs.

This can be very useful for carrying out sensitivity analyses, testing and comparing alternative designs, and asking “what if” questions.

A simple example file which illustrates the use of Scenarios (Scenarios.gsm) can be found in the General Examples/Running folder in your GoldSim directory (accessed by selecting **File | Open Example...** from the main menu).

The details of how to create, run and compare scenarios are presented in the sections below.
**Warning:** Scenarios are an advanced modeling capability. Prior to using this capability, it is strongly recommended that you first learn how to create, run and view model results in the standard way (without scenarios).

**Warning:** Because scenario modeling is somewhat complex (as your model will be storing multiple sets of inputs and results), it is strongly recommended that you read the documentation below before using this feature. The first section (Introduction to Scenarios) is of particular importance. It provides an overview of how the scenario modeling capability works. This is required reading prior to using this capability.

**Note:** Scenario modeling is not supported when doing distributed processing (using the Distributed Processing Module).

### Introduction to Scenarios

GoldSim’s scenario modeling capability allows you to directly compare results generated by different sets of input parameters. In effect, when you use this capability, your model can store (and subsequently compare) multiple sets of results (and inputs):

Because scenario modeling is somewhat complex (as your model will be storing multiple sets of inputs and results), it is strongly recommended that you read the introductory sections below before using this feature. Note that these sections do not provide details on how to create and edit scenarios (that is provided subsequently). Rather, they introduce key concepts and provide an overview of GoldSim’s scenario modeling capability that must be understood before attempting to use this powerful feature.

**What Are Scenarios?**

A scenario is a specific set of input data (and corresponding outputs) for a model. Multiple scenarios can be defined for a model. Different scenarios
within a model are specifically differentiated by having different values for one or more Data elements.

Specifically, a particular scenario is defined by the following:

- A unique Scenario ID (an alphanumeric name);
- An optional description; and
- A set of values for the Data elements in the model for that scenario.

Note that the only elements that can differ between scenarios are Data elements. Of course, not all Data elements in a model will actually differ between scenarios. You typically define a subset of the model’s Data elements as being Scenario Data.

**Read more:** Scenario Data: Defining Inputs for Different Scenarios (page 507).

Once you have defined multiple scenarios, GoldSim allows you to run the model for each scenario and directly compare results generated by each scenario. That is, when you use this capability, your model can store (and subsequently compare) multiple sets of inputs and results.

**Read more:** Running Scenarios and Comparing Scenario Results (page 510).

GoldSim provides several ways to create scenarios.

The most straightforward way to create scenarios is to use the Scenario Manager. The Scenario Manager can be accessed from the main menu (Run | Scenario Manager…) or by pressing F7.

**Read more:** Creating and Editing Scenarios Using the Scenario Manager (page 513).

The Scenario Manager dialog looks like this:

Using this dialog, you can create new scenarios, and delete existing ones. When you add a new scenario, you are prompted for a Scenario Name and Description. In the example shown above, three scenarios have been defined.

Of course, simply creating and naming a scenario is of no value if you do not specify the Scenario Data for each scenario (i.e., the specific Data elements that differentiate the scenarios). As discussed below, this can also be done via the Scenario Manager.

In addition to using the Scenario Manager to create scenarios, once you are experienced with modeling scenarios, you will likely also create scenarios “on the fly” outside of the Scenario Manager using other methods provided by GoldSim. You can even create scenarios via a Dashboard.
Scenario Data: Defining Inputs for Different Scenarios

Read more: Creating and Editing Scenarios in Dashboards (page 527).

Scenario Data are the Data elements that differentiate the various scenarios in your model. GoldSim provides several ways to specify Scenario Data.

The most straightforward way to specify Scenario Data is to use the Scenario Manager. The Scenario Manager can be accessed from the main menu (Run \[ Scenario Manager...\]) or by pressing F7. Within this dialog, you will find a button labeled Scenario Data.... Pressing this button displays the following dialog:

Each row in this dialog represents a Scenario Data element. Each column represents a scenario. You can add new Scenario Data using the Add Element... button. You can subsequently edit the value for any Scenario Data element in any defined scenario. This table is very convenient because it displays all Scenario Data values for all defined scenarios side-by-side.

Read more: Creating and Editing Scenario Data Using the Scenario Manager (page 515).

You can also edit Scenario Data directly in your model. Once you have defined at least one Scenario in your model, the dialog for Data elements is modified slightly. In particular, the dialog provides an option to define a Data element as a Scenario Data element:
In this particular example, we are viewing a Data element that has not yet been defined as being Scenario Data (and hence it has the same value for all Scenarios). As can be seen, a **Scenario Data** checkbox is available (and in this case, unchecked). Checking this box would add this to the list of Scenario Data elements displayed in the Scenario Data dialog accessed via the Scenario Manager.

Here is the dialog for a Data element that has already been defined as Scenario Data:

Note that in this case, the **Scenario Data** box is checked. Like all Data elements, a Scenario Data element has a single value. That is, it cannot display the value for all scenarios; it can only show you the value for one scenario at a time. The dialog indicates which scenario is being displayed (in this example, the “Dry” scenario is being displayed).

As discussed below, you can control which scenario you are viewing in your model by selecting the **Active Scenario**.

**Note:** In addition to editing Scenario Data directly in your model as described above, you can also link Scenario Data elements to Dashboard controls and edit (and create) Scenarios via a Dashboard.

**Read more:** [Creating and Editing Scenarios in Dashboards](#) (page 527).

To highlight those Data elements that are Scenario Data, they have different symbols in the graphics pane:

![Scenario_Data](#)  ![Data](#)

*For Scenario Data, the pencil is orange instead of green.*
After you have defined scenarios in your model, you can browse the model and view Data elements that have been defined as Scenario Data. Note, however, that like all Data elements, a Scenario Data element has a single value. That is, it cannot display the value for all scenarios; it can only show you the value for one scenario at a time. The dialog indicates which scenario is being displayed (in the example below, the “Dry” scenario is being displayed):

Because this is a Scenario Data element, some of the editing is locked (e.g., you cannot change the Type or dimensions). However, in Edit Mode you can change the value itself. If you were to do so, it would change the value of this Data element only for the Active Scenario (in this case, “Dry” scenario).

The Active Scenario is the scenario that is being viewed when you are browsing a model (and hence is only applicable if the model has scenarios defined). That is, whenever a model has scenarios, one of the scenarios is the Active Scenario. The Active Scenario is indicated in the right-hand side of the GoldSim status bar:

If you click on this portion of the status bar, GoldSim will display all of the scenarios (and you can select one to become the Active Scenario):

If you change the Active Scenario and then view a Scenario Data element, it will display the value for the selected scenario.

Note that one of the options here is always the “Live Model”. Live Model can be thought of as a “scratch” scenario, or a temporary placeholder scenario where
you can experiment before saving something as an actual scenario. As will be discussed below, when comparing scenario results, only defined scenarios are shown (the Live Model is not included in the comparison).

You can also select the Active Scenario directly from within the Scenario Manager:

The real power of GoldSim’s scenarios capability is being able to run and view the results of different scenarios in a single plot or table. In order to do this, after defining scenarios, you need to run the different scenarios and compare their results.

It is important to understand two key concepts:

- Scenario results can only be compared when the model is in Scenario Mode. Scenario Mode is a special model state (that is different from Result Mode) that allows scenario results to be displayed and compared.
  
  Read more: Understanding Simulation Modes (page 498).

- When in Scenario Mode, scenario results can only be viewed in Time History Result elements and Distribution Result elements. No other results are available. Hence, if your model does not contain any Time History or Distribution Result elements, you will not be able to compare scenario results.
  
  Read more: Creating and Using Result Elements (page 570).

The most straightforward way to run a scenario and place your model in Scenario Mode is through the Scenario Manager:
You will note that from this dialog, you can run a single scenario (via one of the Run Scenario buttons) or run all scenarios (via the Run All button). In either case, GoldSim will run the model for the single scenario (or all scenarios), and place the model in Scenario Mode.

Here is the same dialog after pressing the Run All button:

Doing so actually runs the model four times in this case (once for each scenario). The dialog indicates that all four scenarios now contain scenario results (the Action column provides the option to Delete Results, which would delete the scenario results for that scenario).

**Warning:** Warning messages are not displayed when running scenarios via the Scenario Manager (although fatal error messages are). To see any warning messages for a scenario, you would need to run it separately (outside of the Scenario Manager) by selecting it as the Active Scenario and pressing F5 or the Run button.

**Read more:** Viewing Detailed Scenario Outputs in Result Mode (page 525).

After running these scenarios, the status bar changes to orange and indicates that the model is in Scenario Mode (the cursor also becomes orange):

Clicking on the right portion of the status bar shows all four scenarios (and the Live Model):
Creating, Running and Comparing Scenarios

Note that all the scenarios have an asterisk (*) next to them. An asterisk indicates that the scenario has scenario results available to be viewed.

If you were then to browse the model, you would note the following:

- If you browse to a Scenario Data element, it will display the value for the Active Scenario.
- There is no option to right-click on an element and view its results. Elements do not store results in Scenario Mode.
- If you double-click on a Time History Result element or a Distribution Result element, GoldSim will display results for all scenarios for which scenario results have been generated (as indicated by the asterisk mentioned above).

For example, a Time History Result would look like this:

![Time History Result](image)

Each scenario is shown and indicated in the legend.

**Note:** If the simulation is probabilistic, then when displaying a Time History of scenarios, GoldSim displays a specified statistic for each scenario (e.g., the mean), as defined in the Monte Carlo Result Options dialog.

**Read more:** Controlling Monte Carlo Result Options (page 482).
Creating, Running and Comparing Scenarios

If you press **F4** you can return to Edit Mode (by deleting all scenario results), or you can choose to only delete the scenario results for the Active Scenario (while keeping the others). In this latter case, you will remain in Scenario Mode.

Pressing **F5** will run the Active Scenario and place the model in Result Mode. This allows you to view detailed results for a particular scenario, as all results for the Active Scenario will be available to view. Note that while in Result Mode, you cannot compare scenarios (e.g., Result elements will only show results for the Active Scenario).

**Read more:** Viewing Detailed Scenario Outputs in Result Mode (page 525).

---

**Note:** All scenario results are deleted when you return to Edit Mode. However, any scenario results that exist prior to entering Result Mode are retained while in Result Mode (and can be viewed again if you return to Scenario Mode).

---

Creating and Editing Scenarios Using the Scenario Manager

A scenario is a specific set of input data (and corresponding outputs) for a model. In particular, different scenarios have different values for one or more Data elements. GoldSim’s scenario modeling capability allows you to directly compare results generated by different sets of input parameters. In effect, when you use this capability, your model can store (and subsequently compare) multiple sets of results (and inputs).

**Warning:** Because scenario modeling is somewhat complex (as your model will be storing multiple sets of inputs and results), it is strongly recommended that you read the introductory section “Introduction to Scenarios” before using this feature. It provides an overview of how the scenario modeling capability works, and should be considered required reading prior to using this capability.

**Read more:** Introduction to Scenarios (page 505).

The Scenario Manager provides the primary mechanism for managing the scenarios in your model. The Scenario Manager can be accessed from the main menu (Run | Scenario Manager…) or by pressing **F7**, and looks like this:

Using this dialog, you can create new scenarios, and delete existing ones. In the example shown above, four scenarios have been defined.

Of course, simply creating and naming a scenario is of no value if you do not specify the Scenario Data for each scenario (i.e., the specific Data elements that differentiate the scenarios). This can also be done via the Scenario Manager.
Creating, Running and Comparing Scenarios

Adding Scenarios Using the Scenario Manager

The sections below discuss the Scenario Manager in detail.

From within the Scenario Manager (accessed from the main menu (Run | Scenario Manager…) or by pressing F7) you can add a new scenario by pressing the New… button. When you do so, you are prompted for a Scenario Name and Description:

Note: By default, the scenario Description is the Analysis description specified in the Information tab of the Simulation Settings dialog (which by default is blank).

Read more: Viewing and Editing Model Summary Information (page 485).

After entering a Scenario ID and Description, this information is displayed in the Scenario Manager:

Note: After creating a scenario, you can edit its Name and/or Description by simply clicking in the field.

You can delete existing scenarios using the Delete button, and move the scenarios up and down in the list using the Move Up and Move Down buttons. (Holding the Ctrl key down switches the Delete button to a Delete All button which can be then used to delete all scenarios).

Of course, simply creating and naming a scenario is of no value if you do not specify the Scenario Data for each scenario (i.e., the specific Data elements that differentiate the scenarios). As discussed below, this can also be done via the Scenario Manager.
Note: In addition to using the Scenario Manager to create scenarios, once you are experienced with modeling scenarios, you will likely also create scenarios “on the fly” outside of the Scenario Manager using other methods provided by GoldSim. You can even create scenarios via a Dashboard.

Read more: Creating and Editing Scenarios in Dashboards (page 527).

The Style and Show columns control how and whether a scenario’s results are displayed in charts. The Action column is used to run scenarios (or delete scenario results).

Read more: Running Scenarios and Displaying Scenario Results (page 520).

Scenario Data are the Data elements that differentiate the various scenarios in your model. Although GoldSim provides several ways to specify Scenario Data, the most straightforward way to do so is to use the Scenario Manager (accessed from the main menu by selecting Run | Scenario Manager… or by pressing F7). Within this dialog, you will find a button labeled Scenario Data….

Pressing this button displays the following dialog:

Each row in this dialog represents a Scenario Data element. Each column represents a scenario. You can add new Scenario Data using the Add Element… button. When you do so, GoldSim will display a browser showing all Data elements in the model. After selecting one, it will add it to the list.

Note: Cloned Data elements cannot be specified as Scenario Data.

Read more: Cloning Elements (page 952).

You can edit the value for any Scenario Data element in any defined scenario directly in this table. You cannot enter links into this table (entries must be values, expressions containing only values, or, if the Data is a Condition, True or False).

You can remove a Data element from the list of Scenario Data using the Remove Element button. The Move Up and Move Down buttons allow you to change the order of the Scenario Data elements in the list (simply for the purpose of better organizing the data; the order has no impact on how the data is used).

This table is very convenient because it displays all Scenario Data values for all defined scenarios side-by-side.
Each column in the table (i.e., each scenario) has a **Copy From** button at the top. Pressing this button displays a drop-list of all the other scenarios (as well as the Live Model):

![Scenario Data Table]

Selecting one of these options copies all of the Data values from the selected scenario to the selected column. This provides a convenient way to rapidly change the values for a scenario (e.g., if you want to define a new scenario in which many of the values are the same as in an existing scenario, with a small number of differences).

**Note:** The **Copy From** button is only available if the scenario defined by that column does not have scenario results. If scenario results have been generated for that scenario, a **Delete Results** button is available.

**Read more:** Running Scenarios and Displaying Scenario Results (page 520).

As discussed below, you can also browse into the model and edit Scenario Data directly. Moreover, you can also link Scenario Data elements to Dashboard controls and edit (and create) Scenarios via a Dashboard.

**Read more:** Creating and Editing Scenarios in Dashboards (page 527).

After you have defined scenarios in your model, you can browse the model and edit existing Data elements that have been defined as Scenario Data, or specify additional Data elements as being Scenario Data.

In particular, once you have defined at least one Scenario in your model, the dialog for Data elements is modified slightly. Specifically, the dialog provides an option to define a Data element as a Scenario Data element:
In this particular example, we are viewing a Data element that has not yet been defined as being Scenario Data (and hence it has the same value for all Scenarios). As can be seen, a **Scenario Data** checkbox is available (and in this case, unchecked). In fact, if you click on **Scenario Data** (the label, not the checkbox itself) a dialog will be displayed to indicate that this Data element provides the same value for all scenarios:

Checking the **Scenario Data** box would add this to the list of Scenario Data elements displayed in the Scenario Data dialog accessed via the Scenario Manager.

**Note:** Cloned Data elements cannot provide Scenario Data (i.e., you cannot check the **Scenario Data** box for a cloned element).

**Read more:** [Cloning Elements](#) (page 952).

Here is the dialog for a Data element that has already been defined as Scenario Data (i.e., the **Scenario Data** box is checked):
If you click on **Scenario Data** (the label, not the checkbox itself) a dialog will be displayed indicating the values used for each scenario:

To highlight those Data elements that are Scenario Data, they have different symbols in the graphics pane:

For **Scenario Data**, the pencil is orange instead of green.

Moreover, if you hold your cursor over a Scenario Data element, it indicates that it provides scenario data:

Like all Data elements, within the Data Definition field, a Scenario Data element displays a single value. That is, it cannot display the value for all scenarios; it
can only show you the value for one scenario at a time. The dialog indicates which scenario is being displayed (in the example above, the “Dry” scenario is being displayed).

In this case, the “Dry” scenario is referred to as the Active Scenario. The Active Scenario is the scenario that is being viewed when you are browsing a model (and hence is only applicable if the model has scenarios defined). That is, whenever a model has scenarios, one of the scenarios is the Active Scenario. The Active Scenario is indicated in the right side of the GoldSim status bar:

If you click on this portion of the status bar, GoldSim will display all of the scenarios (and you can select one to become the Active Scenario):

If you change the Active Scenario and then view a Scenario Data element, it will display the value for the selected scenario. Note that one of the options here is always the “Live Model”. Live Model can be thought of as a “scratch” model, or a temporary placeholder model where you can experiment before saving something as a scenario.

You can also select the Active Scenario directly from within the Scenario Manager:

In addition, when viewing Scenario Data via the Scenario Manager (by pressing the Scenario Data… button), the Active Scenario will be marked in bold.

When viewing a Scenario Data element, you will note that some of the editing is locked (e.g., you cannot change the Type or dimensions). However, in Edit Mode you can change the value itself. You cannot, however, enter links (entries must be values, expressions containing only values, or, if the Data is a Condition, True or False).
If you were to change the value, it would change the value of this Data element only for the Active Scenario.

**Note:** You can also change the value of a Scenario Data element while in Scenario Mode if and only if the Active Scenario has no scenario results.

**Running Scenarios and Displaying Scenario Results**

**Read more:** Running Scenarios and Displaying Scenario Results (page 520).

The real power of GoldSim’s scenarios capability is being able to run and view the results of different scenarios in a single plot or table. In order to do this, after defining scenarios, you need to run the different scenarios and compare their results.

The details of how to run and compare scenarios are presented in the sections below.

In models in which scenarios are not being used, there are three modes that the model can be in: Edit Mode, Run Mode, and Result Mode. The model is in Edit Mode while it is being edited, the model is in Run Mode while the simulation is actually running, and the model is placed in Result Mode after results have been generated.

**Read more:** Understanding Simulation Modes (page 498).

When using scenarios in GoldSim, a fourth mode is introduced: Scenario Mode. Scenario Mode is a special model state that allows scenario results to be displayed and compared. *Scenario results can only be compared when the model is in Scenario Mode.*

Although Scenario Mode is similar in some respects to Result Mode (in that both modes are used to view results), there are several key differences:

- When in Scenario Mode, scenario results can only be viewed in Time History Result elements and Distribution Result elements. *No other results are available.* Hence, if your model does not contain any Time History or Distribution Result elements, you will not be able to compare scenario results.

  **Read more:** Creating and Using Result Elements (page 570).

- A model in Scenario Mode has *at least one* scenario with results. However, when in Scenario Mode, not all scenarios necessarily have results (since the scenarios can be run independently).

- Models cannot be edited in Result Mode. However, in Scenario Mode, you can edit Scenario Data elements for scenarios that do not have scenario results (i.e., if the Active Scenario does not have results, Scenario Data for that scenario can be edited).

  **Note:** Although you can edit Scenario Data elements for an Active Scenario that does not have results while in Scenario Mode, you can never edit non-Data elements in Scenario Mode, since these are common to all scenarios, and doing so would invalidate scenario results that existed for other scenarios.

There are two ways to place a model into Scenario Mode:

- Using the Scenario Manager to run one or more scenarios.
Read more: Generating and Deleting Scenario Results Using the Scenario Manager (page 521).

- Transitioning directly from Result Mode to Scenario Mode.

Read more: Transitioning Directly from Result Mode to Scenario Mode (page 526).

The most straightforward way to run a scenario and place your model in Scenario Mode is through the Scenario Manager:

From this dialog, you can run a single scenario by pressing one of the Run Scenario buttons, or run all scenarios via the Run All button. In either case, GoldSim will run the model for the single scenario (or all scenarios), and place the model in Scenario Mode.

Here is the same dialog after pressing the Run All button:

In this case, pressing the button actually ran the model four times (once for each scenario). The dialog indicates that all four scenarios contain scenario results (the Action column provides the option to Delete Results, which would delete the scenario results for that scenario).

Warning: Warning messages are not displayed when running scenarios via the Scenario Manager (although fatal error messages are). To see any warning messages for a scenario, you would need to run it separately (outside of the Scenario Manager) by selecting it as the Active Scenario and pressing F5 or the Run button.
After running these scenarios, the status bar changes to orange and indicates that the model is in Scenario Mode:

Clicking on the right portion of the status bar shows all four scenarios (and the Live Model):

Note that all the scenarios have an asterisk (*) next to them. An asterisk indicates that the scenario has scenario results available to be viewed.

**Note:** A model in Scenario Mode has *at least one* scenario with results. However, when in Scenario Mode, not all scenarios necessarily have results (since the scenarios can be run and results can be deleted independently). Only those scenarios with results are marked with an asterisk (*).

Once you have generated scenario results, there are a number of ways to delete them. From the Scenario Manager dialog itself, you can delete individual scenario results (using the **Delete Results** button for a particular scenario). In addition, when viewing Scenario Data from within the Scenario Manager (via the **Scenario Data**… button), a **Delete Results** button is available for each scenario:

**Note:** Because, by definition, a model in Scenario Mode has *at least one* scenario with results, if you delete all scenario results from the Scenario Manager, the model will immediately return to Edit Mode.

Finally, if you press **F4** while in Scenario Mode (and the Active Scenario and at least one other scenario have results), you will be presented with this dialog:
The default option is to delete the results for the Active Scenario, but remain in Scenario Mode. Otherwise, you can return directly to Edit Mode (and all scenario results are deleted).

If you press F4 in Scenario Mode (and the Active Scenario and at least one other scenario do not have results), the only option presented is to delete all results and return to Edit Mode.

You can also exit Scenario Mode by running the Active Scenario and going directly to Result Mode. In this case, all existing scenario results are retained.

Read more: Viewing Detailed Scenario Outputs in Result Mode (page 525).

Once a model is in Scenario Mode, if you were to browse the model, you would note the following:

- If you browse to a Scenario Data element, it will display the value for the Active Scenario.
- There is no option to right-click on an element and view its results. Elements do not store results in Scenario Mode.
- If you double-click on a Time History Result element or a Distribution Result element, GoldSim displays results for all scenarios for which scenario results have been generated (and the Show button has been checked).

Read more: Viewing Scenario Results in Time History Result Elements (page 622); Viewing Scenario Results in Distribution Result Elements (page 665).

For example, a Time History Result would look like this:
Each scenario is shown and indicated in the legend.

If the simulation is probabilistic, a Distribution Result will look like this, displaying the probability distribution for each scenario:

If the simulation is deterministic, then when displaying a Distribution Result for scenarios, GoldSim displays a histogram showing a single result for each scenario:
Note: Only Time History and Distribution Result elements display scenario results. Scenario results cannot be displayed for other types of Result elements.

Note: Array results cannot be compared in Scenario Mode. Only scalar results can be displayed. Hence, if an array is listed as one of the results in the Result Properties page, it will not be available for selection in the display dialogs when comparing scenarios (i.e., when the model is in Scenario Mode). If you wanted to compare one or more items from an array in Scenario Mode, you would need to add those scalar array items as separate results in the Result Properties dialog.

If you press F4 you can return to Edit Mode (by deleting all scenario results), or you can choose to only delete the scenario results for the Active Scenario (while keeping the others). In this latter case, you will remain in Scenario Mode.

Pressing F5 will run the Active Scenario and place the model in Result Mode. As discussed below, this allows you to view detailed results for a particular scenario.

Read more: Carrying Out a Simulation (Run Mode) (page 499); Viewing Results (Result Mode) (page 504).

Note: All scenario results are deleted when you return to Edit Mode. However, any scenario results that exist prior to entering Result Mode are retained in Result Mode (and can be viewed again if you return to Scenario Mode).

In order to compare scenario results, you must be in Scenario Mode. Moreover, when in Scenario Mode, scenario results can only be viewed in Time History Result elements and Distribution Result elements (and to some extent, in Dashboards). No other results are available.
Often, however, in addition to comparing scenario results, you will want to look at the detailed results for a single scenario. To do so, you must select the scenario of interest as the Active Scenario, and then run the model in the standard way (e.g., by pressing F5). This places the model in Result Mode for the Active Scenario. All results are available to view.

Read more: Carrying Out a Simulation (Run Mode) (page 499); Viewing Results (Result Mode) (page 504).

Note that while in Result Mode, you cannot compare scenarios (e.g., Result elements will only show results for the Active Scenario).

However, any scenario results that existed prior to entering Result Mode are still saved in the file (they are not deleted if you transitioned from Scenario Mode to Result Mode). You can view these again by returning to Scenario Mode. In particular, when you press F4 from Result Mode, you will be presented with an option to return to Scenario Mode (where you will have access to all scenario results), or return to Edit Mode (which will delete all scenario results).

Read more: Transitioning Directly from Result Mode to Scenario Mode (page 526).

When you are in Result Mode, you can press F4 to delete the detailed output results and return to Edit Mode.

Read more: Understanding Simulation Modes (page 498).

Note, however, that when you press F4, you will also be offered the opportunity to enter Scenario Mode. The options available are a function of whether any scenarios have been previously defined, whether the Active Scenario is the Live Model, and whether any scenario results already exist.

If no scenarios have been previously defined (or if the Active Scenario is the Live Model and no scenario results exist for other scenarios), the following dialog will be displayed when you press F4:

The first option is to delete all results and return to Edit Mode. If you select the second option, you will be prompted for a scenario name (and description), and the model will be placed in Scenario Mode. Although detailed outputs are deleted, the scenario results are saved.

If scenarios have been defined, no scenario results exist, and the Active Scenario is not the Live Model, the dialog displayed is somewhat different when you press F4:
Creating, Running and Comparing Scenarios

The default option is to delete the full results, but enter Scenario Mode (and keep the scenario results). The second option is to delete all results and return to Edit Mode.

If scenario results already exist in your model, and the Active Scenario is not the Live Model, the dialog displayed when you press F4 provides three options:

The default option is to delete the full results, but enter Scenario Mode (and keep the scenario results). The second option is to delete all results, as well as the scenario results for the Active Scenario, and enter Scenario Mode. The final option is to delete all results and return to Edit Mode.

Finally, if scenario results already exist in your model, and the Active Scenario is the Live Model, the dialog displayed when you press F4 provides two options:

The first option is to delete all results, and switch to Scenario Mode. If you select the second option, you will be prompted for a scenario name (and description), and the model will also be placed in Scenario Mode. Although detailed outputs are deleted, the scenario results are saved. If you wish to return all the way to Edit Mode, you must first return to Scenario Mode, and then press F4 again.

You can create, edit and run scenarios from a Dashboard. In particular, Dashboards provide a Scenario control that can be added that allows you to: 1) select the Active Scenario, 2) run the scenario; and 3) add and delete scenarios:

Creating and Editing Scenarios in Dashboards
When a Scenario control is added to a Dashboard, you can enter Scenario Mode and compare scenario results directly from the Dashboard.

Dashboards and the Scenario control are discussed in detail in the GoldSim Dashboard Authoring Module User’s Guide.

Running an Optimization

GoldSim provides the ability to carry out a special type of run to facilitate optimization of your model. For this type of run, you specify an objective function (a specific result that you would like to minimize or maximize), an optional constraint (a condition that must be met), and one or more optimization variables (variables in your model that you have control over).

GoldSim then runs the model multiple times, systematically selecting combinations of values for each of the optimization variables. By doing so, GoldSim can determine the values of the optimization variables that optimize (minimize or maximize) the objective function while meeting the specified constraint.

Typical uses of optimization include:

- Finding the best input data values for a model, in order to match observed historical data (i.e., calibration).
- Selecting the “best” option from among alternatives. “Best” could mean safest, cheapest, most reliable, or another appropriate measure.
- Optimizing the timing of actions or policy changes during the course of a simulation.

The details of how to set up and run an optimization are presented in the sections below.

Two optimization examples can be found in the General Examples/Running folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu): 1) a very simple example file which illustrates the use of optimization for the purpose of calibration (CalibrationOptimization.gsm); and 2) a fairly complex example that illustrates the use of optimization to optimize a design (DesignOptimization.gsm).

In order to run an optimization in GoldSim, you first need to select an objective function that is to be maximized or minimized. The objective function can be any output in your model. The optimization is carried out based on the final value (the value at the end of the realization) of the objective function.

The objective function can be a simple computed measure (e.g., impact, cost) or a complex expression (e.g. “stakeholder satisfaction” or “risk-weighted net present value”). In most cases, it will represent either a cumulative value (e.g., total accumulated cost), a peak value (e.g., the highest water level observed during the simulation), or a valley (e.g., the minimum amount of money in an account during the simulation). Cumulative values can be computed using Integrators, Reservoirs and Pools in GoldSim, while peaks and valleys can be computed using Extrema elements. As a result, the objective function is often a function of one or more of these types of elements.

Read more: Integrator Elements (page 228); Reservoir Elements (page 235); Pool Elements (page 252); Extrema Elements (page 276).

After you have defined the objective function, you need to identify one or more optimization variables (Data or Stochastic elements in your model) that can be adjusted by GoldSim to optimize the objective function. Optimization variables...
often represent “decision variables” in your model – variables that you have
direct control over (e.g., how much money to spend, when to implement
something, the properties of a facility that you are simulating). GoldSim requires
you to specify an initial value (an initial guess) for each variable, as well as an
upper and lower bound between which the variable will be allowed to range.
You can also specify whether the variable is to be limited to integer values.

Obviously, the objective function should be dependent (either directly or
indirectly) on all of the optimization variables.

You can also define a constraint that must be met when selecting optimization
variables (such as not exceeding a regulatory limit, staying within a financial
budget, or never having a combination of optimization variables that would be
unacceptable for some reason). GoldSim will seek the optimal value of the
objective function by varying the optimization variables within their bounds and
will dismiss any solutions which do not meet the specified constraint condition.

Once the objective function and optimization variables have been defined, and
any required constraints specified, the optimizer can be run. The optimization is
based on Box’s complex method.

**Note:** Box’s complex method is described in: Box, M.J. (1965). “A new method
of constrained optimization and a comparison with other methods,” *The

Box’s complex method begins by developing an initial “complex,” which is a
set of valid solutions that meet all of the requirements specified by the user.
This complex contains twice as many valid solutions as the number of
optimization variables. Once the initial complex has been developed, the
algorithm searches the solution space iteratively, replacing the least optimal
members of the complex with more optimal ones until the solution converges or
GoldSim determines that convergence cannot be achieved (in which case a
warning message will be displayed).

**Read more:** Setting the Optimization Precision (page 534); Understanding
Optimization Warning Messages (page 537).

When you complete an optimization run, GoldSim provides options for copying
the optimal values of the optimization variables to the element definitions before
carrying out any further simulations.

To prepare to carry out an optimization, select Run | Optimization… from the
main menu (there is also an Optimization button in the Advanced toolbar).
The following dialog will be displayed:
Defining the optimization settings requires three steps:

- Specifying the objective function and any constraints;
- Specifying the optimization variables; and
- Setting the optimization precision.

These three steps are described in the sections below.

The first step in preparing to carry out an optimization is to define an objective function that is to be maximized or minimized. The objective function can be any output in your model. The optimization is carried out based on the final value (the value at the end of the realization) of the objective function.

In most cases, the objective function will represent either a cumulative value (e.g., total accumulated cost), a peak (e.g., the highest water level observed during the simulation), or a valley (e.g., the minimum amount of money in an account during the simulation). Cumulative values can be computed using Integrators and Reservoirs in GoldSim, while peaks and valleys can be computed using Extrema elements. As a result, the objective function is often a function of one or more of these types of elements.

Read more: Integrator Elements (page 228); Reservoir Elements (page 235); Extrema Elements (page 276).

You can also define a constraint that must be satisfied by the solution. GoldSim will seek the optimal value of the objective function by varying the optimization variables within their bounds and will dismiss any solutions which do not meet the specified constraint condition.
The “Nature of Optimization Problem” section of the Define Optimization Settings tab is used to specify the objective function and any constraints:

To select the **Objective Function** you must click on “<Click to select>”. A browser dialog for selecting an output will be displayed. After you select the output and press **OK**, the selected output is then shown in the dialog:

You can delete the selected output by pressing the red X to the right, or you can click on the item and select a different output.

The **Objective Function** must be a scalar value.

The **Goal** drop-list allows you to select whether you want to maximize or minimize the objective function. The **Precision** drop-list allows you to control the precision of the optimization algorithm.

*Read more:* Setting the Optimization Precision (page 534).

You can specify a constraint by checking the **Required Condition** checkbox and specifying a condition that must be true in order for a solution (i.e., a combination of values of the optimization variables) to be valid. The constraint must be a condition output or a conditional expression, and can be a function of the objective function, the optimization variables, and/or any other outputs in your model:

The **Required Condition** is evaluated at the end of each realization. Solutions that do not meet the specified condition will be dismissed, and no solutions that violate this constraint will ever be included in the complex that is developed by the optimizer.

After you have defined the objective function, you need to identify one or more optimization variables that can be adjusted by GoldSim as it seeks to optimize the objective function. Optimization variables often represent “decision variables” in your model – variables that you have direct control over (e.g., how much money to spend, when to implement something, the properties of a facility that you are simulating) GoldSim requires you to specify an initial value (an initial guess) for each variable, as well as an upper and lower bound between which the variable will be allowed to range.
Obviously, the objective function should be dependent (either directly or indirectly) on all of the optimization variables.

The bottom half of the Define Optimization Settings tab allows you to specify an arbitrary number of optimization variables:

You can add optimization variables by pressing the Add… button.

When you do so, a browser dialog will be displayed for selecting an output. Optimization variables must be scalar Stochastic elements or Data elements. They can be values or conditions.

The dialog will only list Stochastic elements and Data elements (no element types other than Stochastics and Data elements will be listed). It will list Data and Stochastic elements that are arrays, but if you select one, it will display an error message.

**Note:** A clone can be selected as an optimization variable, but two “sister” clones (clones of each other) cannot both be selected. If you try to do so, GoldSim will display an error message.

**Note:** Only scalars can be specified as optimization variables. If you wish to optimize a member or members of a vector or matrix, you can do so by creating individual scalar Data elements that define the members of the vector or matrix.

**Note:** If a Stochastic is selected as an optimization variable, the Probability_Density and Cumulative_Probability outputs for the element in the model will always evaluate to 0 during the optimization runs.

After selecting an element as an optimization variable, the following dialog will be displayed:
The **Name**, **Description**, **Display Units** and **Type** are automatically transferred from the linked element (and cannot be edited here). If you’ve selected the wrong element, you can reopen the browser for selecting optimization variables by clicking on the Name.

For each optimization variable, you must specify the range over which the variable will be varied during the optimization by defining a **Lower Bound** and an **Upper Bound**. The bounds must be numbers and cannot be links to other elements in the model.

You are also required to specify an **Initial Value** for each optimization variable. This should be your “initial guess” for the optimal value of each variable. The Initial Value is not critical, but if it is chosen well (i.e., if it is relatively close to the true optimum), it could speed up the optimization. The Initial Value must be a number and cannot be a link to other elements in the model.

If you check the **Restrict this variable to integer values only**, during the optimization process, GoldSim will only use integer values for the variable.

**Note:** During the optimization process, GoldSim completely overrides the values of the elements chosen as optimization variables. That is, the manner in which they are defined (e.g., the distribution type for a Stochastic optimization variable) has no impact on the optimization.

**Note:** During the optimization process, each run is treated as a deterministic simulation. Hence, all Stochastic elements not defined as optimization variables use their their deterministic value.

Once you have specified bounds and an initial value, you can click **Prev** or **Next** to view the other optimization variables, but the new optimization variable will not be added until you click the **OK** button.

Optimization variables can be removed by highlighting them in the dialog and pressing the **Remove** button.
You can edit the settings for an existing optimization variable by double clicking on the optimization variable in the dialog, or by highlighting the optimization variable and pressing the **Edit...** button.

The optimization algorithm can be controlled via the **Precision** setting in the **Define Optimization Settings** tab:

You can set the **Precision** to Maximum, High, Medium or Low (the default setting is Medium). The precision setting has two effects:

1. it determines the number of valid points used to generate the initial complex of solutions; and
2. it determines the convergence criterion GoldSim uses to decide whether or not it has obtained an “optimal” solution.

Box’s complex method begins by developing an initial “complex,” which is a set of valid solutions that meet all of the requirements specified by the user. This complex contains twice as many valid solutions as the number of optimization variables. Once the initial complex has been developed, the algorithm searches the solution space iteratively, replacing the least optimal members of the complex with more optimal ones until the solution converges or GoldSim determines that convergence cannot be achieved (in which case a warning message will be displayed).

The way in which the initial complex is generated depends on the **Precision** specified by the user:

- For Low Precision, GoldSim generates only the minimum number of valid solutions required to create the complex (i.e., two times the number of optimization variables).

- For Medium Precision, GoldSim generates twice as many valid solutions as it needs (i.e., it generates 4N valid solutions, where N is the number of optimization variables). It then uses the 2N most optimal solutions to form the initial complex.

- For High or Maximum Precision, GoldSim generates five times as many valid solutions as it needs (i.e., it generates 10N valid solutions, where N is the number of optimization variables). It then uses the 2N most optimal solutions to form the initial complex.

Hence, the higher the **Precision** setting, the better the initial complex spans the variable space (and the higher the likelihood that a global optimum, rather than a local optimum, will be found).

GoldSim determines whether or not it has achieved an optimal solution by comparing the range of objective function values in the current complex with the range of objective function values in the initial complex. The convergence criterion is also determined by the **Precision** specified by the user:

- For Low Precision, convergence requires the range in objective function values to be less than 0.01 of the initial range or 100 solutions have been tried.
• For Medium Precision, convergence requires the range in objective function values to be less than 0.001 of the initial range or 1000 solutions have been tried.

• For High Precision, convergence requires the range in objective function values to be less than 0.00001 of the initial range or 1E4 solutions have been tried.

• For Maximum Precision, the optimizer will continue until it can no longer improve the result, or 1E6 solutions have been tried.

In some cases, the optimization may not be able to converge within the allowed number of iterations. In this case, GoldSim will display a warning message.

Read more: Understanding Optimization Warning Messages (page 537).

The Randomize Optimization Sequence checkbox (which is checked by default), can be used to ensure that the initial complex is randomized every time it is created. This can be used to help ensure that GoldSim finds a global optimum by running the optimization more than once.

Read more: Finding a Global Optimum in Complex Models with Multiple Optima (page 537).

Once you have specified the objective function, constraints and optimization variables, the optimization can be run. To run the optimization, simply click the Optimize! button at the bottom of the Optimization dialog.

Note: During the optimization process, each run is treated as a deterministic simulation. You cannot run an optimization if your Monte Carlo settings are specified to run multiple realizations. The settings must be specified to either run a deterministic simulation or a single probabilistic simulation (in which case it will automatically be treated as a deterministic simulation).

Clicking the Optimize! button will immediately take you to the other tab in the optimization dialog (Run Optimization) and begin the optimization. The Run Optimization tab tracks the progress of the optimization:
The **Run Optimization** tab is divided into two sections:

**Best Function Value vs. Iterations**: This section of the dialog provides a graph that plots the most optimal value of the objective function (on the y-axis) versus the number of iterations that have been completed (on the x-axis).

**Top Results**: This section displays the objective function and optimization variable values from the ten most optimal iterations.

This tab is continuously updated during the course of an optimization.

While the optimization is running, a **Pause** button will be available. Pressing it will temporarily stop the optimization (and switch the **Pause** button into a **Resume** button, which can be clicked to restart the simulation). There is also an **Abort** button, which immediately ends the optimization.

After the optimization is completed, the **Pause** and **Abort** buttons will switch back to **Optimize!** and **Close** buttons.

When the optimization completes, a message will be displayed indicating that the optimization has completed. The first row in the Top Ten Results section will display the optimal values of the optimization variables and the optimized value of the objective function. When you press **Close** to exit the optimization, you will be presented with a dialog with options for saving these results.

**Read more**: [Saving Optimization Settings and Results](#) (page 538).

In some cases, the optimization may not converge. In this case, GoldSim will provide a warning message.

**Read more**: [Understanding Optimization Warning Messages](#) (page 537).

Some types of models may have multiple local optima. In such a case, GoldSim may not necessarily find the global optimum. There are, however, some steps you can take to increase the likelihood of finding the global optimum.
Warning: Warning messages are not displayed when running an optimization (although fatal error messages are).

Warning: When running an optimization, Interrupt messages are never displayed. Moreover, unless the Specify Action When Message is off or cannot be shown field in the Interrupt is set to “Continue” or “Skip remainder of current Realization and Continue” (or the Interrupt is disabled), any triggered Interrupts will result in a fatal error.

Read more: Interrupting a Simulation (page 409).

For a well-behaved problem with a single optimum solution, GoldSim can be counted on to locate the optimum. For more complex problems with multiple local optima, however, the solution found by GoldSim may not necessarily be the true optimum (it could be a local optimum).

For complex models with local optima, the choice of bounds and initial values for the optimization variables can be important, and can determine whether GoldSim converges to a local optimum, rather than to the global optimum.

The best way to convince yourself that you have found a global, rather than a local, optimum, is to run the optimization multiple times with different initial complexes.

In order to ensure that GoldSim creates a different initial complex every time it starts an optimization, the Randomize optimization sequence checkbox should be checked (which is the default). If you then run multiple optimizations (by pressing the Optimize! button repeatedly), GoldSim will rerun the optimization with a different initial complex each time.

Note: If the Randomize optimization sequence checkbox is cleared, each optimization will use the same initial complex and hence will converge to the same solution.

Although it is impossible to be sure that you have found a global optimum, if you repeat the optimization multiple times (randomizing the initial complex) and converge to the same solution, your confidence that you have found a global optimum will certainly be increased.

In some cases, an optimization may not converge. In this case, GoldSim will provide one of three warning messages:

Unable to create valid initial complex. Optimization failed. In order to start the optimization, GoldSim must find 2N valid solutions, where N is the number of optimization variables. This error indicates that GoldSim could not find 2N valid solutions. This probably indicates that your problem is over-constrained. That is, GoldSim could not find any combination of optimization variables (within the bounds specified) that met the specified constraint (the Required Condition).
Cannot improve on best solution, but failed to converge. **Optimization is probably successful.** This indicates that the optimization has found a number of valid solutions, but is “stuck” and cannot find any better solutions (while still failing to meet the convergence criterion). In this case, the solution is likely a good one, and it simply indicates that the convergence criterion was too strict. You should carefully examine the Top Results to determine if the optimal values are still significantly changing.

Solution still slowly improving, but failed to converge. **Optimization is probably successful.** This indicates that the optimization has found a number of valid solutions, and continues to slowly improve the solution, but has not converged after a large number of iterations (100 for low, 1000 for medium, 10,000 for high precision). In this case, the solution is likely a good one, and it simply indicates that the convergence criterion was too strict. You should carefully examine the Top Results to determine if the optimal values are still significantly changing.

**Optimizing a Probabilistic Model**

If you wish to optimize a probabilistic (uncertain) system, the objective function to be optimized cannot be a single deterministic output. Rather, it must be a statistic. That is, if X was an output of a probabilistic model (and hence was output as a probability distribution), optimizing X itself would be meaningless. Rather, you would need to optimize a particular statistic (e.g., the mean or 50th percentile) of the output X.

In order to do this within GoldSim, you must use SubModels. In particular, you must embed a SubModel within an outer model. The SubModel would be a fully dynamic Monte Carlo simulation, and the outer model would be a static optimization. The optimization variables for the outer model would be statistics that have been exposed on the output interface of the SubModel.

**Read more:** Using SubModels to Embed Models Within Models (page 973).

**Saving Optimization Settings and Results**

When you close the Optimization dialog, you will see a dialog with a number of options:

At the top of the dialog are two radio buttons that determine whether the optimization settings (i.e., all of the settings on the **Define Optimization Settings** tab) are saved. If the first radio button is selected (Save all optimization variables and settings), all of these settings are saved. If the second radio button is selected, all of these settings are deleted (and hence will not be available the next time you want to run an optimization).

Even if you save the optimization settings, the optimization results in the **Run Optimization** tab will be deleted and will not be available the next time you run an optimization (you will need to re-run the optimization). Hence, before
exiting the Optimization dialog, you should record the optimal values of the optimization variables and the optimized value of the objective function displayed in the Top Ten Results section of the Run Optimization tab.

**Note:** GoldSim provides an easy way to extract and save the Top Results. Left-click on a column and then drag the cursor to select the other columns you wish to export. Alternatively, press **Ctrl+A** to select the entire Top Results table. Pressing **Ctrl+C** copies it to the clipboard (with the column headers). It can then be pasted into another application.

This information (the optimization results) can also be copied directly to other parts of the model. The bottom portion of the dialog provides two options for doing so (you can do neither, one or both).

The options are as follows:

**Copy optimized values to initial values of variables for next optimization:** This option replaces the Initial Values that you specified for each optimization variable with the values that resulted in the most optimal value of the objective function. This is useful, for example, if you plan to modify the model in the future, and think that the optimal values from the original model would be a good initial guess for the modified model. Note that this option is grayed out if optimization settings and variables are not saved.

**Copy optimized values to element definitions:** This option replaces the values of the optimization variable elements in the GoldSim model with the values that resulted in the most optimal value of the objective function. For optimization variables that are Data elements, the element definition is replaced with the optimized value. For optimization variables that are Stochastic elements, the “specified value” for deterministic simulations is replaced with the optimized value.

**Read more:** [Specifying a Deterministic Value for a Stochastic](page 180).

### Running Sensitivity Analyses

GoldSim provides the ability to carry out a special type of run to facilitate sensitivity analyses. For this type of run, you specify the result you are interested in, and one or more variables that you want to analyze (which must be Stochastics or Data elements).

GoldSim then runs the model multiple times, systematically sampling each variable over a specified range, while holding all of the other variables constant. This then allows GoldSim to produce sensitivity plots (i.e., a tornado chart and X-Y function charts) to assist you in graphically identifying the variables in your model to which the results is most sensitive.

**Note:** The sensitivity analyses discussed here produce graphical (and tabular) outputs created by changing one variable at a time, while holding all other variables constant. GoldSim also provides a second type of sensitivity analysis in which statistical sensitivity measures are computed by analyzing the results of multiple realizations of the model where all of the Stochastic variables are simultaneously sampled each realization.
Running Sensitivity Analyses

Read more: Viewing a Sensitivity Analysis Table (page 681).

Note: During the sensitivity analysis, each run is treated as a deterministic simulation. Hence, all Stochastic elements not defined as independent variables use their determinitic value.

The details of how to set up, run and view such a sensitivity analysis are presented in the sections below.

A simple example file which illustrates the use of sensitivity analysis (Sensitivity.gsm) can be found in the General Examples/Running folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

To prepare to carry out a sensitivity analysis, select Run | Sensitivity Analysis... from the main menu (there is also a Sensitivity Analysis button in the Advanced toolbar). The following dialog will be displayed:

Selecting the Result and Independent Variables for a Sensitivity Analysis

Note: You can only run a sensitivity analysis from Edit Mode. You cannot run a sensitivity analysis if the model is in Result Mode.

The first step required to set up a sensitivity analysis is to define which output (i.e., result) you would like to analyze for sensitivity. The sensitivity analysis will be used to determine the degree to which other variables impact this result. You select this result by pressing “<Click to select>” immediately to the right of Result to Analyze.

When you do so, a browser dialog will be displayed for selecting an output. You can select any scalar, value output in your model (you cannot select arrays or conditions). After you select the output and press OK, the selected output is then shown in the dialog:
You can delete the selected output by pressing the red X to the right, or you can click on the item and select a different output.

After selecting the result to analyze, you must then select the variables that will be varied to determine the degree to which they impact the result. You add these variables by pressing the **Add…** button.

When you do so, a browser dialog will be displayed for selecting an element. The dialog will only list Stochastic elements and Data elements (no other element types will be listed). You can select any Stochastic or Data element with a scalar, value output (you cannot select arrays or conditions). Each selected independent variable will then be listed in the dialog:

You can delete a variable that you have already added by selecting the variable in the list and pressing the **Delete** button. Pressing the **Add All Stochastics** button adds all valid Stochastic elements (i.e., Stochastics that are scalar values) in your model to the variable list.

**Note:** If a Stochastic is selected as an independent variable for a sensitivity analysis, the Probability_Density and Cumulative_Probability outputs for the element in the model will always evaluate to 0 during the sensitivity analysis runs.

You can also delete one variable and replace it with another by selecting the variable name. When you do so, a browser dialog will be displayed for selecting a new element.
Once you have specified a result and added one or more independent variables, the Sensitivity Analysis dialog will look similar to this:

When GoldSim carries out a sensitivity analysis, it does so by running a series of deterministic simulations, varying one independent variable at a time through a range of values. You select the range of values using the Lower Bound, Central Value and Upper Bound fields in this dialog.

Read more: Deterministic Simulation Options (page 482).

For each simulation, GoldSim must select single values for all of the other independent variables (that are not currently being varied), and for any other Stochastic elements in the model. The manner in which these single values are selected is as follows:

- For Stochastics that are identified as one of the independent variables, GoldSim uses the specified Central Value for the single value.
- For Stochastics that are not listed as one of the independent variables, GoldSim uses the same value it would use if it were carrying out a deterministic simulation.
- For Data elements that are identified as one of the independent variables, GoldSim uses the specified Central Value for the single value.
- For Data elements that are not listed as one of the independent variables, GoldSim uses its defined value.

In order to carry out a sensitivity analysis, you must specify the range for each variable, as well as the number of points to sample within each range.

The number of points sampled for each variable is specified in the #Points field. This field defaults to 11, and must be an odd number.

For Data variables, you must always specify the Lower Bound, Central Value and Upper Bound directly. You enter numbers (without units). The units are displayed to the left of these three fields.
For Stochastic variables, you have a choice of how to define the range. If Use quantiles for Stochastic elements is checked (the default), GoldSim defines the bounds and the Central Value by using quantiles of the distribution. The Central Value is always the 50th percentile. The Lower and Upper Bounds are determined by the #Points using the following equations:

\[
\text{Lower Bound} = \frac{100\%}{(\text{# Points}/2)} \\
\text{Upper Bound} = 100\% - \text{Lower Bound}
\]

For example, with 11 points, the Lower Bound is approximately 4.5% and the Upper Bound is approximately 95.5%.

If Use quantiles for Stochastic elements is cleared, you must always specify the Lower Bound, Central Value and Upper Bound directly for Stochastics, just as you would for Data elements.

**Warning:** When defining the Lower Bound, Central Value and Upper Bound directly for Stochastics or Data elements, these values do not have to be related in any way to the actual definition for the elements. For example, you could specify a Lower Bound, Central Value and/or an Upper Bound that were outside the range of the actual defined distribution for a Stochastic. Similarly, a Data element could be defined such that its actual defined value was different from its Central Value when running Sensitivity Analyses. In general, this is likely to lead to confusion, and it is strongly recommended that when defining the ranges for your independent variables, the values should be consistent with the actual defined values for the elements that are used when not running a Sensitivity Analysis.

**Viewing Sensitivity Analysis Results**

Once you have specified a result, added one or more independent variables, and defined the ranges for the independent variables, you can carry out several different types of sensitivity analysis.

GoldSim provides four types of analysis and display:

- Central Value Result;
- Tornado Chart;
- X-Y Function Chart; and
- Result Data Display.

These are discussed in the sections below.

**Note:** The sensitivity analyses discussed here produce graphical and tabular outputs created by changing one variable at a time, while holding all other variables constant. GoldSim also provides a second type of sensitivity analysis in which statistical sensitivity measures are computed by analyzing the results of multiple realizations of the model where all of the Stochastic variables are simultaneously sampled each realization.

**Read more:** Viewing a Sensitivity Analysis Table (page 681).
Note: During the sensitivity analysis, each run is treated as a deterministic simulation. You cannot run a sensitivity analysis if your Monte Carlo settings are specified to run multiple realizations. The settings must be specified to either run a deterministic simulation or a single probabilistic simulation (in which case it will automatically be treated as a deterministic simulation).

Warning: Warning messages are not displayed when running a sensitivity analysis (although fatal error messages are).

Warning: When running a sensitivity analysis, Interrupt messages are never displayed. Moreover, unless the Specify Action When Message is off or cannot be shown field in the Interrupt is set to "Continue" or "Skip remainder of current Realization and Continue" (or the Interrupt is disabled), any triggered Interrupts will result in a fatal error.

Read more: Interrupting a Simulation (page 409).

Sensitivity Analysis: Central Value Result

The simplest type of sensitivity analysis within the Sensitivity Analysis dialog is to compute the Central Value Result. This is a single deterministic simulation in which the specified Central Value is used for all defined independent variables. Pressing the Calculate button causes GoldSim to run a single deterministic simulation, and display the value of the selected result in the Central Value Result field:

This can be a useful way to quickly run a series of “what if” deterministic analyses, by manually changing the Central Values for one or more of the independent variables.

Sensitivity Analysis: Tornado Chart

A tornado chart is a type of sensitivity analysis that provides a graphical representation of the degree to which the Result is sensitive to the specified Independent Variables.

A tornado chart can be produced by pressing the Tornado Chart... button in the Sensitivity Analysis dialog. When you do so, GoldSim runs a series of deterministic simulations, varying one independent variable at a time through a range of values. In particular, it does the following:

1. While holding all other Independent Variables at their Central Value and all other Stochastics (not identified as Independent Variables) at their deterministic values, GoldSim runs 3 deterministic simulations with three different values for the first dependent variable: the Lower Bound, the Central Value, and the Upper Bound.

2. The process is repeated for each Independent Variable.
For example, if there were 10 Independent Variables specified, GoldSim would carry out 30 deterministic simulations to produce the tornado chart.

Once it carries out these simulations, it uses the results to construct a tornado chart such as this:

![Tornado Chart Example](image)

The x-axis of a tornado chart represents the values of the Result for different values of the independent variables.

Each bar represents the range of Result values produced when each independent variable is set to Lower Bound, Central Value, and Upper Bound (with the other variables being held constant). A light blue bar indicates that the value was produced by the Lower Bound (Low), and a dark blue bar indicates that the value was produced by the Upper Bound (High). For example, this particular chart indicates that the variable d produced a Result equal to about 1.25 when d was at its Lower Bound, and a Result of about 0.2 when d was at its Upper Bound.

The variables are organized from top to bottom according to the total range of Results produced. That is, the variable that produces the largest range of the Result between its Lower and Upper Bound is at the top of the chart. Hence, bars become smaller toward the bottom of the chart, and the overall effect is to take on the appearance of a “tornado”.

The solid vertical line represents the value of the Result when the Central Values are used for all independent variables. In this example, the Result using the Central Values is 1.3.

**Note**: If you have more than 20 independent variables, GoldSim will only plot the top 20 variables in the tornado chart.

**Sensitivity Analysis: X-Y Function Chart**

An X-Y function chart is a type of sensitivity analysis that provides a graphical representation of the degree to which the Result is sensitive to the specified Independent Variables.

An X-Y function chart can be produced by pressing the **X-Y Function Chart** button in the Sensitivity Analysis dialog. When you do so, GoldSim runs a...
Running Sensitivity Analyses

A series of deterministic simulations, varying one independent variable at a time through its range of values. In particular, it does the following:

1. While holding all other Independent Variables at their Central Value and all other Stochastics (not identified as Independent Variables) at their deterministic values, GoldSim runs n deterministic simulations, where n is the number specified by #Points. For each simulation, the first Independent Variable is varied from its Lower Bound to its Upper Bound.

2. The process is repeated for each Independent Variable.

Once it carries out these simulations, it uses the results to construct an X-Y function chart such as this:

The y-axis of an X-Y function chart represents the values of the Result for different values of the independent variables.

There is one line for each variable. Each line illustrates how the Result changes when that independent variable is varied from its Lower Bound to its Upper Bound (with the other variables being held constant). The number of points used to create each line is determined by the #Points field.

**Note:** One of the points is always the Central Value. Half of the remaining points are spaced evenly (linearly) between the Lower Bound and Central Value, and the other half of the remaining points are spaced evenly (linearly) between the Central Value and the Upper Bound.

Because each variable likely will have different units and a different range, the x-axis does not represent actual values; rather it represents normalized values (and hence they all range from 0 to 1). The normalization differs depending on whether the range of the independent variable was specified as quantiles or was specified directly.

For Stochastic independent variables whose Bounds have been entered as quantiles, the x-axis represents the actual quantile of each point. For Data elements (and Stochastics) whose Bounds have been specified directly (as
Running Sensitivity Analyses

Lower Bound, Central Value and Upper Bound), the x-axis represents the following:

\[
\text{Normalized Value} = \frac{\text{Value} - \text{Lower Bound}}{\text{Upper Bound} - \text{Lower Bound}}
\]

If you would like to plot a single independent variable (on the x-axis) against the result (on the y-axis), using the actual value of the independent variable rather than the normalized value, you can do so by pressing the appropriate button in the column labeled **Plot** in the Independent Variable table. A chart like this will be displayed:

In some cases, you may simply want to view the raw data used to produce the X-Y function chart. You can do so by pressing the **Result Data...** button in the Sensitivity Analysis dialog. When you do so, a table like this is displayed:

This table shows the actual value for the independent variables (rather than the normalized values).

**Note:** You can copy the contents of this table to the clipboard. To do so, you must first select the entire table (by double-clicking on the empty cell in the upper left-hand corner of the table, or by pressing **Ctrl+A**). After you do so, you can copy the table to the clipboard by **Ctrl+C**. You can subsequently paste the table into another application (such as a spreadsheet).
The Run Log

Whenever you run a GoldSim model, a Run Log is produced. The Run Log contains basic statistics regarding the simulation (e.g., the version of the program, the date, the simulation length), and any warning or error messages that were generated.

If your simulation generates any warning messages during the run, you will immediately be prompted to view the Run Log when the simulation completes. The Run Log can also be viewed at any time after a simulation is completed by selecting Run|View Run Log in the main menu.

**Note:** The Run Log is not produced when running scenarios, sensitivity analysis or optimization.

The Run Log is automatically viewed by whatever application is associated with ".txt" files (typically Notepad or WordPad).

**Note:** The Run Log only becomes available after a model has been run. In Result Mode, the Run Log contains information describing the current run. If a model is in Edit Mode, the Run Log contains information describing the most recent run (the results of which are no longer available).

A simple Run Log is shown below:

![Run Log Example](image)

**Note:** Although the Run Log is viewed in a text editor, it is saved internally within the GoldSim file and cannot be edited. If you edit the Run Log using the text editor, the changes will not be saved in the GoldSim file.
When you view a Run Log, GoldSim creates a text file from the information in the GoldSim file. By default, this file is saved in the same location as the model file. If it cannot be saved there (due to access issues), GoldSim will save it to the user’s temporary folder (and provide the location of the file in a message).

By default, the name of the Run Log file is “GoldSim Run Log.txt”.

**Note:** On rare occasions, you may want to instruct GoldSim to insert the model filename into the name of the Run Log filename. To do this, you must edit the Windows Registry. In particular, add a DWORD registry key under HKEY_CURRENT_USER\Software\GTG\Settings named RunLogEmbedModelName and set it to a non-zero value. If you do so, the name of the Run Log file will be “ModelFilename_Run Log.txt”. For example, if the model filename was called “Example.gsm”, the Run Log file would be named “Example_RunLog.txt”.

The Run Log contains the following information:

- General simulation information, such as the GoldSim version number, the filename, the time the simulation was begun (in effect, a timestamp for the run), the total simulation time, and the Analysis Description.
- All of the key simulation settings, summarizing exactly how the simulation was carried out.
- Any warning and error messages associated with the run. If the run has been aborted, this will also be indicated.
- Whether (and when) any elements have been downloaded from a database prior to running the model.

*Read more:* [Linking Elements to a Database](page 1031).

The Run Log is important because it provides an internal, uneditable "audit trail" for the run. By looking at the Run Log, you can immediately see key information like when the model was run, what version of GoldSim was used, and whether there were any warning messages.

### Running GoldSim from the Command Line

In some cases, it may be useful to bypass GoldSim user interface and run GoldSim directly from the command line.

This, for example, could be done by selecting Run from the Windows Start menu, which displays the following dialog:
You must enter the full path to the GoldSim executable (which may, of course, be different than that shown above), followed by the GoldSim file that you want to open. If no filename is specified, GoldSim opens a new file.

**Note:** If no path is specified for a filename, GoldSim looks in the last directory from which a GoldSim file was opened. Hence, it is highly recommended that you provide a full path for the filename.

The primary reason for running GoldSim from the command line is to take advantage of a number of command line parameters that cause GoldSim to execute in a specific manner.

The command line parameters are summarized in the following table. Note that the parameters are entered after the executable and before the model file name.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>-show name</td>
<td>Opens GoldSim and jumps to the specified Container or Dashboard</td>
<td>Used if you want a GoldSim model to simply open immediately and display the contents of a specific Container or Dashboard. If this parameter is used to reference a Container name that is used multiple times within a model (within localized Containers), GoldSim opens the first Container with the specified name found while searching downward through the containment hierarchy.</td>
</tr>
<tr>
<td>-r</td>
<td>Opens the GoldSim file and immediately runs the simulation.</td>
<td>In this situation, GoldSim automatically skips the “Ready” state and starts the simulation.</td>
</tr>
<tr>
<td>-run</td>
<td>EXECutes a global database download for all elements linked to databases.</td>
<td>The date is only required for Extended GoldSim databases. The date must be specified in local format and be enclosed in quotes (e.g., “7 March 2013”).</td>
</tr>
<tr>
<td>-sv [filename]</td>
<td>Saves the model file using either the optionally provided output filename, or the input filename (if an output file is not provided).</td>
<td>This option can be used to run and save a model, or to open a model and save it in a newer version of GoldSim.</td>
</tr>
</tbody>
</table>
### Parameter Description Comments

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>-s -slave</td>
<td>Opens GoldSim in Slave Mode.</td>
<td>Used when running models on multiple processors using GoldSim’s distributed processing features (GoldSim Distributed Processing Module). This command cannot be used with any other parameters.</td>
</tr>
<tr>
<td>-d “element.input=equation” -data “element.input=equation”</td>
<td>Inserts the provided equation into the specified input for the specified element.</td>
<td>This parameter allows you to change the contents of an input field for any element in GoldSim using the command line. The input can be omitted (only the element name needs to be specified) if the input name is “Definition” (e.g., for Data and Expression elements). This parameter is particularly useful if you want to stack a number of runs in a batch file, with each run changing one or more element definitions.</td>
</tr>
</tbody>
</table>

Examples of how some of these parameters would commonly be used are provided below. Note that these parameter strings would follow the name (and path) of the GoldSim executable (goldsim.exe).

### Command line Comments

<table>
<thead>
<tr>
<th>Command line</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>-x –sv test1.gsm</td>
<td>Opens test1.gsm, saves it, and closes GoldSim. Useful, for example, if you want to convert test1.gsm to the latest GoldSim version.</td>
</tr>
<tr>
<td>-x –sv test2.gsm test1.gsm</td>
<td>Opens test1.gsm, saves it as test2.gsm, and closes GoldSim. Only the destination model (test2 in this case) is saved. Useful, for example, if you want to convert test1.gsm to the latest GoldSim version, but preserve the original file.</td>
</tr>
<tr>
<td>-r test1.gsm</td>
<td>Opens test1.gsm, executes pre-run checks and starts the simulation. GoldSim remains open after the simulation. The model file is only saved if auto-save is enabled in the Options dialog.</td>
</tr>
<tr>
<td>-r –sv test1.gsm</td>
<td>Opens test1.gsm, executes pre-run checks and starts the simulation. At the end of the simulation, the model file is saved. GoldSim remains open after the simulation.</td>
</tr>
<tr>
<td>-r –sv –x test1.gsm</td>
<td>Opens test1.gsm, executes pre-run checks and starts the simulation. At the end of the simulation, the model file is saved. GoldSim is then closed.</td>
</tr>
<tr>
<td>-r –sv test1res.gsm test1.gsm</td>
<td>Opens test1.gsm, executes pre-run checks and starts the simulation. At the end of the simulation, the model file is saved to test1res.gsm. GoldSim remains open after the simulation.</td>
</tr>
</tbody>
</table>
### Command line

<table>
<thead>
<tr>
<th>Command line</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>-slave</td>
<td>Starts GoldSim in Slave mode. This command cannot be used with any other parameters.</td>
</tr>
</tbody>
</table>

Several points should be noted regarding the use of the `–d` (-data) command:

- The argument to this command should be enclosed in quotes.
- Units must be specified if the input has dimensions.
- More than one input can be defined in a command line. Note, however, that Windows limits the length of a command string (to 8191 characters).
- If a Data element is connected to a Dashboard control, the Dashboard will always take precedence. As a result, if you want to control an element via a command line, it should not be connected to a Dashboard control.
- The `–d` parameter will only find global elements. If an element is located within a localized Container, it will not be found.

Examples of the use of the `–data` command are provided below:

<table>
<thead>
<tr>
<th>Command line</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>-d &quot;Data1=5m3&quot;</td>
<td>Puts the value of 5 m3 into the input of element Data1. This would only be valid if Data1 had an input named Definition (e.g., if it was a Data element or Expression element).</td>
</tr>
<tr>
<td>-d “Pond.Initial Value = 10m3”</td>
<td>Puts the value of 10 m3 into the Initial Value input of element Pond (in this case, a Reservoir).</td>
</tr>
<tr>
<td>-d “X = if(etime&gt;5day,1,2)”</td>
<td>Puts the expression <code>if(etime&gt;5day,1,2)</code> into the input of element X. This would only be valid if X had an input named Definition (e.g., if it was a Data element or Expression element).</td>
</tr>
<tr>
<td>-d “Data1=5m3” -d “Data2=5sec”</td>
<td>Puts the value of 5 m3 into the input of element Data1, and 5 sec into the input of element Data2. This would only be valid if Data1 and Data2 had an input named Definition (e.g., if they were Data or Expression elements).</td>
</tr>
</tbody>
</table>

In most cases, it is likely that you will want to “stack” a number of GoldSim runs by creating a batch file that you run from the command line. However, if you simply stack command lines and then run a batch file, all GoldSim calls would be launched in parallel (e.g., if you had 10 command lines that ran GoldSim, 10 instances of GoldSim would open and run simultaneously).

To avoid this problem and force the commands to run in series, you can use the Start/Wait command provided by Windows. An example batch file that utilizes this is shown below:

```
CD c:\Program Files (x86)\GTG\GoldSim 11.1
Start /wait GoldSim.exe -r -sv r1.gsm -d "InputA=10m" -x "c:\runs\input.gsm"
Start /wait GoldSim.exe -r -sv r2.gsm -d "InputA=20m" -x "input.gsm"
Start /wait GoldSim.exe -r -sv r3.gsm -d "InputA=30m" -x "input.gsm"
Start /wait GoldSim.exe -r -sv r4.gsm -d "InputA=40m" -x "input.gsm"
```
The first line simply sets the directory to the location of the GoldSim executable. The next line opens a GoldSim model called input.gsm. By specifying the path here, this sets the path for all input and output files in the remainder of the batch file. GoldSim changes the value of InputA to 10m, runs the model and saves it as r1.gsm. GoldSim then closes. After closing, the next GoldSim command is implemented (open input.gsm again, change InputA to 20m, run and save the model to r2.gsm). This continues until all four GoldSim runs are complete.
Chapter 8: Displaying Results in GoldSim

Often the most effective way to describe, explore, and summarize a set of numbers - even a very large set - is to look at pictures of those numbers.

Edward Tufte, *The Visual Display of Quantitative Information*

Chapter Overview

GoldSim has powerful charting and display functions that allow you to view your simulation results in a variety of ways. You can plot time histories of your data, view probability distributions, create scatter plots and bar charts, and view tables of results. You can also combine multiple results on a single plot, and view multiple plots simultaneously. To facilitate the creation of report quality graphics, you can modify and save chart styles, which allow you to customize (and reuse) the style (i.e., appearance) for each type of chart.

This chapter describes how you can create and view the results of your simulations.

In this Chapter

The following topics are discussed in this chapter:

- Displaying Results: An Overview
- Viewing Time History Results
- Viewing Distribution Results
- Viewing Multi-Variate Results
- Viewing Final Values for Arrays
- Editing the Appearance of a Chart
- Creating and Using Chart Styles
- Exporting Results
- Copying and Exporting Result Displays
Displaying Results: An Overview

The sections below provide the basic information you need to understand the types of result displays you can produce in GoldSim. They also provide references to topics that describe these various result display features in greater detail.

A GoldSim model file is always in one of four modes (or states): Edit Mode, Run Mode, Result Mode, or Scenario Mode.

The mode that the model is in is clearly identified in the status bar (in the lower left-hand corner of the GoldSim window). In addition, the status bar takes on a different color in each mode:

- **This model is in Edit Mode. The status bar is blue.**
- **This model is in Run Mode. The status bar is pink.**
- **This model is in Run Mode, but is paused. The status bar is rust brown.**
- **This model is in Result Mode. The status bar is green.**
- **This model is in Scenario Mode. The status bar is orange.**

**Read more:** [Understanding Simulation Modes](page 498).

Results can be viewed in two modes: Result Mode and Scenario Mode. Result Mode is the most common way that you will view results. Scenario Mode only exists if you are using GoldSim’s scenario feature. Viewing results in Scenario Mode is similar to viewing results in Result Mode, with a number of key limitations.

**Read more:** [Viewing Scenario Results](page 577).

After you run your model (e.g., by pressing **F5**), it will automatically be placed in Result Mode. When in Result Mode, elements with outputs that have been saved are identified in two ways: 1) the elements (and their outputs) are **bold** in the browser (and output interfaces); and 2) the element output ports in the graphics pane are **green**.

When a model is in Result Mode, you can navigate it, open property dialogs, and view result plots. You can also move elements around within the graphics pane, add graphics (e.g., text, images), and add and edit Result elements.

*You cannot, however, edit the model in any way that would change it’s behavior (e.g., you cannot change values in property dialogs or add or delete elements).* GoldSim enforces this rule to ensure that the results and inputs to a model do not become inconsistent.

If you press **F4**, choose **Run|Return to Edit Mode** from the file menu, or press the Edit Mode button in the standard toolbar, the results will be discarded and the model will be returned to Edit Mode.
Although GoldSim provides a variety of powerful ways in which you can display simulation results in the form or charts and tables, the simplest way to view basic results is via tool-tips.

In Result Mode, an output's last Value (the value at the end of the last realization) will be displayed in a tool-tip when the cursor is held over it in a browser (or output interface).

Read more: Using Tool-Tips (page 109).

If the output is the primary output of an element, the last Value will also be shown in the tool-tip displayed when the cursor is held over the element itself.

Tool-tips are always displayed for scalar outputs. If you hold the cursor over a vector or a matrix, only a portion of the last Values may be displayed (if the array is large). Note, however, that if you expand the array in a browser or output interface and hold the cursor over a single item, the last Value for that item will be displayed.

Read more: Using Vectors and Matrices (page 782).

Note: The last Value for an output is displayed in tool-tips regardless of whether results (Time Histories or Final Values) have been saved for the output.

Read more: Saving Outputs as Results (page 494).

Note: You can control the number of significant figures displayed in tool-tips from the Results tab of the Options dialog (accessed via Model | Options… from the main menu).

Read more: The Results Tab of the Options Dialog (page 448).
GoldSim provides four ways in which you can display results in graphical and tabular form:

**Time History Results**: Values at selected times for a particular output.

**Distribution Results**: Probability distributions of the final value (i.e., the value at the end of each realization) of an uncertain output.

**Multi-Variate Results**: Comparisons between the final values of variables (outputs), such as scatter plots, correlation tables, and sensitivity analyses; and

**Array Results**: Final values for vector and matrix outputs.

Right-clicking on an element with a primary output (in the graphics pane or browser) or on a specific output (in an output interface or a browser) which has been saved will provide a context menu for displaying these four types of results.

Depending on the output type and what kinds of results have been saved, up to three of the result options will be available.

**Note**: Time History Results are only available if you have selected **Save Time Histories** for the output. Multi-Variate Results and Distribution Results are only available if you have selected **Save Final Values** for the output. Array Results are only available if the output is a vector or a matrix and you have selected **Save Final Values** for the output.

---

Read more: **Saving Outputs as Results** (page 494).

All four result types can be viewed either as a **chart** or a **table** (and in some cases, there are multiple types of charts and/or tables that can be viewed for a particular type of result).

A chart or table accessed and viewed in this manner is referred to as an **interactive result**. For each type of interactive result, there is a default display that is shown (a chart for all but the Distribution Result, which shows a summary page with both chart and tabular information). Once a result is displayed, you can subsequently switch back and forth between different chart and table views.

Charts or tables can also be viewed using specialized elements referred to as **Result elements**. Result elements allow you to organize all of your key result charts or tables, and allow you to access those results via a simple double-click.

**Read more**: **Creating and Using Result Elements** (page 570).

**Note**: Interactive Time History Results for an output are not available for multi-realization runs unless the output is connected to a Result element.
Examples of the four types of result displays are provided below. Details regarding the manner in which these result displays can be manipulated and customized are provided in subsequent sections.

**Time History Results**

Time History results show the "history" of a particular output as a function of time, and are probably the most common form of result display you will use. A time history chart looks like this:

![Time History Chart Example](image1)

The same information can also be viewed in table form, in which the time is shown in one column, and the output's value is shown in the other:

![Time History Table Example](image2)

**Distribution Results**

In many systems, you may be uncertain about some of the input parameters. In such a case, GoldSim allows you to define these parameters as probability distributions. If at least one of the inputs to a model is a probability distribution, by definition, the outputs are probability distributions. (Basic concepts of probabilistic simulation are provided in Appendix A.) Distribution results provide a way to view the final values of uncertain (probabilistic) outputs.
The summary display for a distribution result (the default display for an interactive result) combines graphical and tabular information, and looks like this:

The chart display for a distribution expands the small chart provided in the upper right-hand corner of the summary display shown above.

Note that distribution charts can be viewed as probability density functions (PDFs), cumulative distribution functions (CDFs), or complementary cumulative distribution functions (CCDFs).

The table display for a distribution shows the final value for each realization:
When evaluating the results of your simulations, you will sometimes want to view multi-variate results, in which multiple outputs are analyzed in graphical or tabular form.

To view multi-variate results, you select a specific output that you are interested in, along with the input variables (which are typically Stochastics) that may have affected that result.

One example of this is the 2-D scatter plot, in which the final value of one output is plotted against the final value of another:

GoldSim also allows you to create a 3-D scatter plot, in which one output is plotted against two others.

GoldSim provides two very useful tabular multi-variate result outputs to support sensitivity and uncertainty analysis. One table provides a variety of statistical sensitivity analysis outputs (coefficient of determination, correlation coefficients, standardized regression coefficients, partial correlation coefficients, and importance measures):

**Multi-Variate Results**
A second table displays a complete correlation matrix, which indicates the degree to which all selected variables are correlated to each other:

![Correlation Matrix](image)

You can also view the raw data display of multiple variables, showing the final values for each realization for each of a number of selected outputs:

![Raw Data Display](image)

**Array Results**

Many complex models will utilize vectors and matrices, collectively referred to as arrays. GoldSim allows you to view array results in both graphical and tabular form.

**Read more**: [Using Vectors and Matrices](page 782).

A chart display of a vector looks like this:
In tabular form, the vector looks like this:

<table>
<thead>
<tr>
<th>Fruit</th>
<th>Production (kg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apples</td>
<td>100</td>
</tr>
<tr>
<td>Oranges</td>
<td>110</td>
</tr>
<tr>
<td>Peaches</td>
<td>85</td>
</tr>
<tr>
<td>Pears</td>
<td>140</td>
</tr>
</tbody>
</table>

One form of a chart display for a matrix looks like this:

In tabular form, the matrix is displayed as follows:
Although GoldSim allows you to view four different types of results (time histories, distributions, multi-variate results, and array results), the result display windows for these result types share a number of common characteristics:

- All four results can be viewed either as a chart or a table (and in some cases, there are multiple types of charts and/or tables that can be viewed for a particular type of result). For interactive results, the default is to show a chart (except for the Distribution Result, which shows a summary page with both chart and tabular information). Once a result is displayed, you can subsequently switch back and forth between different chart and table views.

- Although you can resize and maximize an interactive chart or table result display window, it is always modal. That is, it cannot be minimized, and with the exception of viewing the result properties dialog (which can be displayed simultaneously and share the focus with the result display window), it retains the focus while it is displayed. As a result, you must close the result display window before you can edit any other part of your GoldSim model.

  Read more: Viewing and Editing Result Properties (page 564).

- GoldSim does, however, provide a mechanism (Result elements) by which you can display one or more modeless result display windows (which allow you to navigate and edit other parts of the GoldSim model while keeping the windows open).

  Read more: Creating and Using Result Elements (page 570).

- Chart and table result display windows for all four types of results have a common set of buttons which provide access to specialized dialogs and/or allow you to carry out specific actions.

- You can control the number of significant figures displayed in result displays from the Results tab of the Options dialog (accessed via Model | Options… from the main menu).

  Read more: Controlling Significant Figures and Scientific Notation in Result Displays (page 568).

- Charts for all four result types provide a number of common functions, such as context-sensitive menus, data tips and the ability to zoom in on a section of the chart.

  Read more: Using Context Menus in Charts (page 566); Viewing Data Tips in Charts (page 567); Zooming in on a Chart (page 567).

For each type of interactive result, there is a default display that is shown (a chart for all but the Distribution Result, which shows a summary page with both chart and tabular information). Once a result is displayed, you can subsequently switch back and forth between different chart and table views.
Whenever you are viewing a result display, you can choose to view the properties for that particular result. All result display windows have a toolbar at the top. This toolbar contains buttons, a number of which are common to all result display windows. For all result display windows, the furthest button to the right is for displaying the Result Properties dialog:

Pressing this button displays the Result Properties dialog for the result. Note that when you press this button, it does not close the result display window you were displaying. Both windows are displayed simultaneously (although you can subsequently close one or the other). When viewing results, any changes you make to the Result Properties are immediately represented in the result display.

The Result Properties dialog is always modal. That is, it cannot be minimized, and with the exception of viewing the result display (which can be displayed simultaneously and share the focus with the Properties dialog), it retains the focus while it is displayed. As a result, you must close the Result Properties dialog before you can edit any other part of your GoldSim model.

Result Properties differ somewhat for each of the four basic result types.

Read more: Viewing the Properties of a Time History Result (page 581); Viewing the Properties of a Distribution Result (page 640); Viewing the Properties of a Multi-Variate Result (page 675); Viewing the Properties of an Array Result (page 690).

The Properties for a Distribution Result are shown below, and the attributes that are shared among all result types are highlighted:

The following attributes are common to Result Properties dialogs:

- At the top of the page is a button to Create Element. By pressing this button, a Result element is created. When you do so, the key change that will be noticeable is that the Name will now be editable. The Create Element button becomes an Appearance button (which is used to modify the appearance of the element itself).
Displaying Results: An Overview

Read more: Creating and Using Result Elements (page 570).

- The Show Result >> button opens a result display window (while keeping the Properties dialog simultaneously open).
- The Close button closes the Properties dialog.

In addition, all Result Properties dialogs (except for Arrays) allow you to add additional results to the display (i.e., multiple outputs) using the Add Result… button, and jump directly to the selected result using the Go to Result >> button. You can also modify the Label for each result (which can subsequently be displayed in legends and headers, footers and axes labels).

Using Context Menus in Charts

All result charts in GoldSim have context menus from which you can carry out a variety of actions. By right-clicking anywhere in a chart, the following context menu is available:

<table>
<thead>
<tr>
<th>- CHART -</th>
</tr>
</thead>
<tbody>
<tr>
<td>View</td>
</tr>
<tr>
<td>Reset Scale</td>
</tr>
<tr>
<td>Edit Chart Style…</td>
</tr>
<tr>
<td>Copy</td>
</tr>
<tr>
<td>Export…</td>
</tr>
</tbody>
</table>

The options in this menu are as follows:

- **View**: Expands to show three options (Show Header, Show Footer, Show Legend). This allows you to toggle these on and off.

- **Reset Scale**: GoldSim allows you to zoom within a chart (by dragging your cursor from one point of the chart to another while holding down the Ctrl key). Reset Scale allows you to reset to the original scale.

Read more: Zooming in on a Chart (page 567).

- **Edit Chart Style…**: This provides access to a dialog for editing the chart style.

  Read more: Editing the Appearance of a Chart (page 702).

- **Copy**: Copies the chart to the clipboard (from where it subsequently can be pasted into another application).

  Read more: Copying a Chart or Table (page 734).

- **Export…**: Exports the chart to a selected graphics format.

  Read more: Exporting a Chart (page 734).

In addition to the Chart context menu, by right-clicking on a specific item (the Header, Footer or Legend), you can access a context menu that pertains just to that item:

<table>
<thead>
<tr>
<th>- HEADER -</th>
<th>- FOOTER -</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paste</td>
<td>Paste</td>
</tr>
<tr>
<td>Hide</td>
<td>Hide</td>
</tr>
<tr>
<td>Edit Header</td>
<td>Edit Footer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>- LEGEND -</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hide</td>
</tr>
<tr>
<td>Edit Legend</td>
</tr>
</tbody>
</table>

These menus can be used to Hide the item, edit it directly (by going directly to the appropriate tab on the Chart Style dialog), or, in the case of Headers and Footers, paste text.
GoldSim allows you to view a "data tip" by holding the cursor over a data point in a chart. The example below shows a data tip for a point in a time history chart:

In this particular case, the data tip displays the realization number for the curve, and X and Y values of the data point. (For a time history chart, the X axis is always time).

In some situations, you may want to zoom in on a portion of a chart that is of particular interest. You can, of course, edit the style of the chart and change the range of the axes which are displayed.

Read more: Editing the Appearance of a Chart (page 702).

GoldSim also provides a more direct way to zoom in on a section of a chart. In particular, if you drag your cursor from one point of the chart to another while holding down the Ctrl key you will create a selection box. Releasing the left mouse button zooms in on the selected portion of the chart. To return to the original scale, right-click on the chart, and select Reset Scale from the chart context menu, or press Shift+R.

In some cases, you may want to select, copy and paste values from a result table into another application (e.g., a spreadsheet).

Selecting rows or columns in a result table is straightforward. You can select a row or column by left-clicking in any row or column label:

Contiguous rows or columns can be selected by dragging the mouse while it is pressed. Non-contiguous rows or columns can be selected by holding the Ctrl key down while left-clicking. The entire table can be selected by clicking on the cell in the upper left-hand corner of the table.
Once a selection has been made, it can be copied to the clipboard by pressing \texttt{Ctrl+C}.

All result tables can be sorted in ascending or descending order. To sort a table, simply select a column (by clicking in it) and press the Sort button:

![Sort selected column](image)

When you do so, a context menu will be displayed allowing you to sort in ascending or descending order:

![Sort options](image)

After selecting one of these options, that column then becomes the sort key for the table (all columns are sorted based on that column). Whenever the data is sorted, the header of the sort key is shown in red:

![Sorted table](image)

\textit{In this case, the second column is the sort key, and the header is shown in red. In addition, the Sort button changes to indicate the direction of the sort (in this case, descending).}

You can choose a new sort key by simply clicking on a different column and pressing the Sort button again.

Note that the Sort button changes its appearance when sorting is on. The button appears depressed, and one of the two arrows appears larger, and points in the direction of decreasing values (i.e., if the sort is in descending order the larger arrow points down; if it is in ascending order, the larger arrow points up).

To turn sorting off, press the Sort button again and select Reset Sorting.

\textbf{Controlling Significant Figures and Scientific Notation in Result Displays}

GoldSim provides a number of options for controlling the way that Results are displayed. These include the minimum number of significant figures to use, whether or not scientific notation is used, how conditions are displayed and how currencies are displayed. These options are controlled via the Results tab of the Options dialog (accessed via Model | Options\ldots from the main menu).


Read more: The Results Tab of the Options Dialog (page 448).

In addition, you can control the minimum number of significant figures to use and the magnitude above which scientific notation is used if directly from chart and table displays using accelerator keys:

Minimum number of significant figures to display: This setting can be changed dynamically from within most result displays using Alt-Left and Alt-Right. The default value is 4.

Use scientific notation if absolute value is >=: This setting can be changed dynamically from within most result displays using Alt-Up and Alt-Down. The default value is 6 (and GoldSim will always use scientific notation if the magnitude of the value is less than 0.0001 or greater than 1e10).

Note: When labeling chart axes, GoldSim respects the specified scientific notation setting, but ignores the significant figures setting (significant figures in chart axes are determined automatically and cannot be user-controlled).

Size of Values Displayed

Although internal calculations are carried out using double precision numbers, results are only stored as single precision numbers (in order to reduce storage requirements). As a result, result values can only be relied upon to a maximum of 7 significant figures. For date outputs (e.g., outputs of the Milestone element), dates displayed in and around 2000 are only accurate to approximately 2 minutes.

This also means that when results are viewed in tables or charts, the range of values that can be displayed is between -1.2E-38 and 3.4E38.

Viewing Results at Capture Points

Distributions, Multi-Variate results, and Array results all operate on Final Values. That is, they allow you to view the value at the end of each realization.

In some cases, however, you may also want to capture these results at other times in the simulation (rather than just the end of the simulation). This can be useful, for example, if you wanted to view a result distribution for an output at various times during a simulation. GoldSim facilitates this by allowing you to create Capture Points at which these results are also made available.

Read more: Creating Capture Points for Final Value Results (page 469).

If you have created Capture Points, an additional drop-list (labeled “Time”) is added to the result display windows for Distribution, Multi-Variate, and Array results:
To view the results for a particular Capture Point, you simply select it from the drop-list.

A simple example file illustrating Capture Points (CapturePoints.gsm) can be found in the can be found in the General Examples/ Running folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

**Note:** Several elements (Stochastics, SubModels, Spreadsheet elements) produce a special type of complex output representing all the distribution information (Distribution output). This output can be viewed in a Distribution Result. However, when doing so, Capture Points are ignored (when viewing a Distribution output, Capture Points don’t apply, since a Distribution output contains a single “snapshot” and cannot represent multiple “snapshots”).

### Creating and Using Result Elements

While the interactive result display procedure (in which you right-click on an element or an output and select a result display option from a context menu) is appropriate in many cases, it has several limitations:

- It requires several steps to actually view the result (expand an element's outputs, right-click on an output, select display options).
- The outputs you want to display may be scattered across various portions of your model, forcing you to navigate through the model to find and display the results.
- The result display is modal, meaning you can only view one display at a time.

To address these limitations, GoldSim allows you to create **Result elements**. A Result element is a specialized element representing a particular result display. As such, there are four types of basic Result elements: Time History, Distribution, Multi-Variate, and Array. When you double-click on a Result element, the result is directly displayed:
Result elements have several important advantages:

- They allow you to access results via a simple double-click.
- They allow you to make key results readily accessible by placing them in one or more convenient locations (e.g., in a single "results" container) to facilitate presentations and use of the model by others.
- All the result characteristics are saved with the Result element, so that you can completely control and customize how the result will be displayed when it is subsequently viewed (e.g., in a presentation or by another user). Once you display the result, the last size and position of the result display is also saved with the Result element.
- Result elements can be opened prior to the start of a simulation, and are updated dynamically as the simulation progresses (e.g., as each realization is completed).
- The display windows for Result elements are modeless, meaning that you can view multiple display windows simultaneously.

In addition, in some cases, Result elements have special capabilities that do not exist for the corresponding interactive result.

Read more: Specialized Uses for Result Elements (page 576).

### Adding Result Elements to Your Model

There are two ways to create a Result element: by converting an interactive result into a Result element, or by creating a Result element directly.

While viewing an interactive result display (in which you right-click on an element or an output and select a result display option from a context menu), you can convert it to a Result element.
You can do so from the Result Properties dialog for the result you are viewing. 

**Read more:** [Viewing and Editing Result Properties](page 564).

This dialog can be accessed by pressing the Result properties button while viewing the chart or table. This will display the Result Properties dialog:

![Result Properties dialog](image)

*This is the Result Properties dialog for a Distribution result. Dialogs for other result types are similar.*

If you press the **Create Element** button, GoldSim will insert an element. The **Name** will also become editable (a default name will be automatically inserted that you can change). You will also note that the **Create Element** button changes to **Appearance**:

![Appearance button](image)
Note: With one exception, Result elements follow the same rules as other GoldSim elements regarding their names (e.g., can only include letters, numbers and the underscore character). Unlike other elements in GoldSim, however, the Name of a Result element can contain spaces.

For some result types, some additional fields may also become available when the Result element is created (in the example above, an Options... button becomes available).

You can also insert a result element into your model directly just as you would insert any other kind of element, via the main menu or context menus:

Result elements are similar to other elements in GoldSim, in that you can move them, copy them and change their appearance. There are, however, several important differences:

- When you double-click on a result element in Result Mode, Run Mode or Scenario Mode, the result window is immediately displayed. You can display the (empty) result window of a Result element directly in Edit Mode, by Ctrl-double-clicking it (double-clicking on a Result element in Edit Mode displays its properties dialog).

- Result elements do not have input or output ports (since they do not have any outputs). Although they do reference other elements (the results being displayed), these are not considered inputs. Hence, there are no influence lines drawn to Result elements:

A result element has the same result display and editing capabilities as an interactive result display. You can choose to view a table or chart form of the result, or view the Result Properties page (e.g., in order to add other outputs to the display). Result elements “remember” the last type of result format (i.e., table or chart), and display that format the next time they are viewed (double-clicked).

Like an interactive result, you can edit the appearance of the chart (e.g., headers, footers, etc.).

Read more: Editing the Appearance of a Chart (page 702).
Note, however, that whenever you change the appearance of the chart for a result element, *these changes are automatically saved with the element.*

One of the key features of result elements is that they can be opened prior to the start of a simulation (via a Ctrl+double-click), and then viewed during the simulation. The display is updated whenever the simulation is paused, and at the end of every realization. If you are running a single realization, Time History results are updated continuously as the simulation proceeds.

This can be particularly useful in helping you to debug a model (by pausing the simulation at specific points).

The dynamic display of result elements is facilitated by GoldSim’s ability to slow down the speed of the simulation, step through a simulation one realization or timestep at a time, and pause and resume a simulation.

*Read more:* [Pausing and Stepping through a Simulation](page 502).

Note that you cannot open a result element while a model is running. If you want to view a result element being dynamically updated, you must open it prior to starting the simulation, or while the simulation is paused. If you open a result element prior to starting the simulation and try to view a chart, the resizable result display window will open with a message that there is no data available.

Unlike other elements in GoldSim, when you reference an output of an element within a Result element, it is not treated as a “link”. That is, the Result element is not considered to be a “function of” of the outputs that it references (since the Result element is not actually carrying out a calculation; it is simply serving as a mechanism to display a result). As a result, you will note that influences are not drawn between the element(s) being referenced and the Result element. Moreover, the Result element does not appear in the “Affects” list for the element being referenced.

*Read more:* [Understanding Influences](page 98); [Viewing Element Dependencies](page 112).

Nevertheless, it is often necessary to quickly find the elements being referenced by a Result element (and conversely, the Result element that references a particular element). To facilitate this, GoldSim provides a number of tools for browsing between Result elements and the outputs (and hence elements) that they reference:

- From within the Properties dialog for a Result element, if you select a result (i.e., an output), and then press the **Go to Result>>** button, the dialog will close and the output’s element will be selected (i.e., GoldSim will jump directly to that location in the graphics pane). (Note that Array results do not have a **Go to Result>>** button, as this button is most valuable when a Result element contains multiple results).

- From within the Properties dialog for a Result element, if you place your cursor over the result, a tool-tip will be displayed showing the location of the element being referenced:
If you Ctrl+double-click on the result, the dialog will close and the output’s element will be selected (i.e., GoldSim will jump directly to that location in the graphics pane).

If you double-click on the result, a browser will be shown illustrating the location of the element in the hierarchy:

- If your right-click on a Result element in the graphics pane (or the browser), a context menu will be displayed listing all of the referenced outputs:
If you click on one of the outputs listed, the output’s element will be selected (i.e., GoldSim will jump directly to that location in the graphics pane).

- If you right-click on a referenced element in the graphics pane (or the browser), a context menu will be displayed listing all of the Result elements that reference outputs from the element:

  - Time history results for simulations with multiple realizations can only be displayed for outputs that are referenced by Time History Result elements.

    Read more: Viewing Time Histories of Multiple Realizations (page 597); Saving Outputs as Results (page 494).

- Time History and Distribution Result elements can be used to display Scenario results.

    Read more: Viewing Scenario Results in Time History Result Elements (page 622); Viewing Scenario Results in Distribution Result Elements (page 665).

- Time History Result elements can be specified to export time history results (i.e., model outputs) to spreadsheets and text files automatically at the end of a simulation.
Read more: Exporting from a Time History Result Element to a Spreadsheet (page 720); Exporting from a Time History Result Element to a Text File (page 727).

- Time History Result elements can be specified to plot results based on Reporting Periods.
  Read more: Viewing Reporting Period-Based Results in Time History Result Elements (page 607).

- Time History Result elements within a parent model can be used to display time history results from inside a SubModel.
  Read more: Viewing SubModel Results in Time History Result Elements (page 617).

- Distribution Result elements within a parent model can be used to display nested Monte Carlo simulation results generated using a SubModel.
  Read more: Carrying Out Nested Monte Carlo Simulation Using a SubModel (page 992).

- Time History Result elements can be specified to display results at unscheduled update points.
  Read more: Viewing Unscheduled Updates in Time History Result Elements (page 629).

Viewing Scenario Results

GoldSim’s scenario modeling capability allows you to directly compare results generated by different sets of input parameters. In effect, when you use this capability, your model can store (and subsequently compare) multiple sets of results (and inputs).

Read more: Creating, Running and Comparing Scenarios (page 504).

In particular, when a model is in Scenario Mode, if you were to browse the model, you would note the following:

- There is no option to right-click on an element and view its results. Elements do not store results in Scenario Mode.

- If you double-click on a Time History Result element or a Distribution Result element, GoldSim displays results for all scenarios for which scenario results have been generated (and for which the Show button has been checked from within the Scenario Manager).

That is, scenario results can only be viewed in Time History and Distribution Result elements.

Read more: Viewing Scenario Results in Time History Result Elements (page 622); Viewing Scenario Results in Distribution Result Elements (page 665).

Classifying and Screening Realizations

When carrying out probabilistic simulations, you may often run hundreds or thousands of realizations. In order to analyze the results, it is often quite useful to classify the realizations into categories. A category is simply defined by a condition relating one or more outputs in the model (e.g., those realizations in which the discount rate was above 3.5%; those realizations in which the profit exceeded $1,000,000; those realizations in which the peak concentration was between 1 mg/l and 10 mg/l).

The power of categories is that once you have defined them, you can use them in two ways:
• Within certain kinds of result charts (e.g., scatter plots, time histories, distributions), realizations from each category can be displayed in a different color (and/or symbol).

• Within all result displays, you can choose to screen out one or more categories, so that the results that are shown (in charts and tables) only include those realizations in the categories which you have chosen to include.

Classification categories are defined at the bottom of the Monte Carlo Result Display Properties dialog:

This dialog can be accessed from the Monte Carlo tab of the Simulation Settings dialog (via the Result Options… button), and is also accessible from result properties dialogs.

Read more: Viewing and Editing Result Properties (page 564).

By default, all results are placed in a single category (defined as “All realizations”). You can add categories by pressing the Add button. When you do so, it adds a row above the selected row. For each category that you add, you must define a Label and a Condition.

In this example, three categories have been defined:

It is critical to understand how the categories of realizations are created by GoldSim. In particular, the order of the Classification Conditions is important. If multiple Classification Conditions are true for a realization, the realization is assigned to the category with the first True Condition in the list. Hence, in the example shown above:

• The “Low” category consists of all realizations in which $X < 0.2$. As indicated by the Net % column, 11% of the realizations fall into this category.

• The “Medium” category consists of all realizations in which $X >= 0.2$ (and hence they do not fall into the “Low” category above it) AND $X < 0.6$. As
indicated by the Net % column, 46% of the realizations fall into this category.

- The “High” category consists of all remaining realizations (all realizations in which the conditions in the first two categories are false). In this case, it implies that the category consists of all realizations in which X >= 0.6. As indicated by the Net % column, 43% of the realizations fall into this category.

As pointed out above, the Net % columns indicates the percentage of realizations falling into each category. The Gross % column shows the percentage of realizations falling into the selected category and all the categories above it.

**Note:** If you place your cursor over the Net % field, a tool-tip will display the actual realizations that fall into that category; if you place your cursor over the Gross % field, a tool-tip will display the actual realizations that fall into that category and all the categories above it.

**Note:** The Gross % and Net % are only shown when the model is in Result Mode (or when the model is paused).

### Read more: Understanding Result Mode (page 556).

You can use the Move Up and Move Down buttons to change the order of the Conditions (and hence the definition of the categories).

Several points should be noted regarding defining the categories:

- You cannot edit the Condition for (or move) the final category. It always represents “everything else”. That is, it represents all realizations that do not fall into one of the previous categories.

- The Condition can be a complex expression, but it must be a scalar condition.

- You can edit the Conditions in Edit Mode, Result Mode or Run Mode (but not Scenario Mode). However, in Result Mode or Run Mode, you cannot reference an output for which Final Values have not been saved.

Categories are powerful because in Result Mode, within certain kinds of result charts (e.g., scatter plots, time histories, distributions), realizations from each category can be displayed in a different color (and/or symbol). For example, this Multi-Variate Result (a 2D scatter plot) breaks down the results by category:
Note that for each category, you can select a **Style**. This is used when plotting the results when you have created multiple categories. The manner in which these Styles are used is a function of the type of result being plotted.

**Read more:** [Using Result Classification and Screening in Time History Results](page 615); [Using Result Classification and Screening in Distribution Results](page 658); [Using Result Classification and Screening in Multi-Variate Results](page 686); [Using Result Screening in Array Results](page 699).

In addition, you can choose to screen out one or more categories of realizations by clearing the **Include** checkbox for a category:

<table>
<thead>
<tr>
<th>Include</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>Medium</td>
</tr>
<tr>
<td>✔️</td>
<td>High</td>
</tr>
</tbody>
</table>

When you do so, when showing results (in both charts and tables), only those realizations in the categories which you have chosen to include are displayed.

**Note:** You can edit the **Include** box in Edit Mode, Result Mode Scenario Mode or Run Mode.

Obviously, if you have screened out results in such a way, it is critical that this is made apparent that the results being shown are screened. Hence, when results are screened, this is indicated in the right-hand corner of the status bar. A hashtag (and a comment) are added after “Results”:

```
# If some Monte Carlo result categories are screened
```
GoldSim has powerful charting capabilities that allow you to customize the appearance of each chart (e.g., by adding headers and footers, changing axes scales and labels, etc.).

Read more: Editing the Appearance of a Chart (page 702).

In some cases, after modifying the appearance of a chart, you may want to save the particular set of properties you have created (e.g., label fonts, header, footer) so that you can apply them to another chart. To facilitate this, GoldSim allows you to save the set of properties as a named chart style. The named chart style is saved as part of the model file, and can subsequently be applied to other results in your model. You can also export and import chart styles between models.

In order to facilitate the use of a single chart style for multiple results, GoldSim allows you to use keywords (delimited by the % symbol) when you create and use chart styles. For example, you could define a chart style in which the header was %rname%. This particular keyword inserts the name of the Result element you are displaying. Similarly, the keyword %curdate% inserts the current date.

When you view a result, it will initially take on a set of characteristics defined by the default chart style for that particular type of chart (e.g., time history, distribution, etc.). GoldSim automatically provides a set of default chart styles for all chart types (that utilize the keywords). If desired, however, for any type of chart, you can specify one of your own custom chart styles as the default chart.

Read more: Creating and Using Chart Styles (page 712).

Viewing Time History Results

Time History results show the “history” of a particular output as a function of time, and are probably the most common form of result display you will use.

A Time History result has two types of views:

- Chart View;
- Table View.

If you have saved Time Histories for an output, you can display a Time History result by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element's primary output) and selecting Time History Result... from the context menu. By default, a chart will be displayed.

The specific time points that are displayed in Time History results are determined by your simulation settings.

Read more: Specifying When Time History Results Will Be Saved (page 464).

Note: Time history results for simulations with multiple realizations can only be displayed for outputs that are referenced by Time History Result elements.

Viewing the Properties of a Time History Result

The topics below discuss the display and use of Time History results in detail.

If you click on the Result Properties button when viewing a Time History Result chart or table, the Properties dialog for the result will be displayed. The Result Properties button is the furthest button to the right at the top of the display window:
Viewing Time History Results

Note that when you press this button, it does not close the result display window you were displaying. Both windows are displayed simultaneously (although you can subsequently close one or the other). When viewing results, any changes you make to the Result Properties are immediately represented in the result display.

The Result Properties dialog is always modal. That is, it cannot be minimized, and with the exception of viewing the result display (which can be displayed simultaneously and share the focus with the Properties dialog), it retains the focus while it is displayed. As a result, you must close the Result Properties dialog before you can edit any other part of your GoldSim model.

The Result Properties dialog for an interactive (scalar) Time History result looks like this:

You can show the full path for a result (showing the containment hierarchy) by placing your cursor over the Result item to display a tool-tip.

At the top of the page is a button to Create Element. By pressing this button, a Result element is created. When you do so, the key change that will be noticeable is that the Name will now be editable. The Create Element button becomes an Appearance button (which is used to modify the appearance of the element itself).

Read more: Creating and Using Result Elements (page 570).

The Show Result >> button opens a result display window (while keeping the Properties dialog simultaneously open).

Read more: Viewing a Time History Chart (page 584); Viewing a Time History Table (page 586).
The **Add Result…** button allows you to add results to the chart. Results can be deleted with the **Delete Result** button. The order in which multiple results appear in the list can be changed using the **Move Up** and **Move Down** buttons.

**Read more:** [Viewing Time Histories of Multiple Outputs](page 588).

For each result in the list, you specify a **Style** (used in charts), and the **Label** used in legends in charts and headers in tables.

**Read more:** [Controlling the Chart Style in Time History Results](page 635).

---

**Note:** The colors used for a Probability chart (which shows the percentiles for a multi-realization simulation) are not specified by the **Style** button. Rather, these are controlled via the Global Monte Carlo Result Options.

**Read more:** [Viewing Probability Histories for Multiple Realizations](page 600).

The **Y1** and **Y2** checkboxes are used to determine which axis the result is plotted on. If you add a second output that has the same dimensions as the original output, it will automatically be placed on the primary (Y1) axis. If it has different dimensions, it will automatically be assigned to the secondary (Y2) axis. You can add as many outputs as desired to the list. If you add an output whose dimensions do not match those currently assigned to the Y1 or Y2 axes, it will be assigned to neither. As a result, it will not be displayed in the chart (but will be displayed in the table). If you specifically try to assign an unassigned output to an axis, any outputs assigned to that axis with different dimensions will be cleared and the axis will take on the new dimension.

You can also specify the **Display Units** for the Y1 and Y2 axis (which overrides the display units specified within the element’s property dialog).

The Properties dialog for a Time History Result element has several differences:
• The **Create Element** button is replaced by an **Appearance** button (which is used to modify the appearance of the element itself).

• The **Name** (and **Description**) are editable (since this is an element).

• An **Options...** button is available that provides access to the Monte Carlo Result Display Properties.  
  *Read more:* [Controlling Monte Carlo Result Options](page 482).

• Time History Result elements also have two additional options at the bottom of the dialog: the ability to disable the element, and the ability to export results (to a spreadsheet or text file) at the end of a simulation.  
  *Read more:* [Disabling a Time History Result Element](page 635); [Exporting From a Time History Result Element to a Spreadsheet](page 720); [Exporting from a Time History Result Element to a Text File](page 727).

• The **Time Display Setting** can only be edited while in Edit Mode. It defaults to “Simulation Time”, which is the appropriate setting for viewing standard time histories. However, two additional options are also available (“Reporting Periods” and “SubModel Time”) which can be selected in order to view Reporting Period-based results and SubModel results, respectively, in Time History Result elements.  
  *Read more:* [Viewing Reporting Period-Based Results in Time History Result Elements](page 607); [Viewing SubModel Results in Time History Result Elements](page 617).

If you have run multiple realizations, the Properties dialog for a Time History Result dialog has one additional field that is not available for single realization runs: you can specify a **Custom Statistic** for each output:

![Time History Result Elements](image)

This can then be viewed in result displays.  
*Read more:* [Viewing Custom Statistics for Multiple Realizations](page 603).

If you have saved Time Histories for an output, you can display a Time History chart by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element's primary output) and selecting **Time History Result...** from the context menu. By default, a chart will be displayed.

---

**Note:** When viewing a Time History Result element, the element “remembers” the last type of view that was displayed, and displays that view when you double-click on it.

**Note:** Time history results for simulations with multiple realizations can only be displayed for outputs that are referenced by Time History Result elements.

If you are viewing a table rather than a chart, you can view a chart by pressing the **Chart** button at the top of the display.
A Time History Chart looks like this:

![Time History Chart](image)

The Time History Chart has a variety of buttons and controls at the top of the window. Some items are always present, while others are only shown under certain circumstances (e.g., when viewing multiple realizations and/or multiple outputs).

The following buttons are always available:

- **Display Table**: Selecting this button switches to a Time History Table view of the result. Note that when viewing a table, the Display Table button appears selected; when viewing a chart, the Display Chart button appears selected.

- **Edit Chart Style**: This button provides access to a dialog for editing the chart style.

  ![Chart style](image)

  **Read more:** [Controlling the Chart Style in Time History Results](page 635).

- **Edit Properties**: This provides access to the Result Properties.

  ![Edit Properties](image)

  **Read more:** [Viewing the Properties of a Time History Result](page 581).

In some situations, you may want to zoom in on a portion of a chart that is of particular interest. You can, of course, edit the style of the chart and change the range of the axes which are displayed. GoldSim also provides a keyboard shortcut to support this.

**Read more:** [Zooming in on a Chart](page 567).

Condition outputs are either True or False and are typically used as state variables or flags in a simulation. As a result, in some situations, you may want to save and plot time histories of conditions. To facilitate this, when plotting a time history of a condition, GoldSim plots True as 1 and False as 0.
Like all result charts, you can also control various attributes of the chart via a context menu. This includes the ability to turn on and off a legend. The legend uses the Label for the result defined in the Result Properties page. When viewing multiple results on one chart, you will always want to display the legend.

**Read more:** Using Context Menus in Charts (page 566); Viewing Time Histories of Multiple Outputs (page 588).

You can control the number of significant figures displayed in result displays from the Results tab of the Options dialog (accessed via Model | Options… from the main menu).

**Read more:** Controlling Significant Figures and Scientific Notation in Result Displays (page 568).

You can copy the chart to the clipboard by right-clicking in the chart, and selecting Copy from the context menu. Alternatively, you can press Ctrl+C while viewing the chart. To export a chart, right-click in the chart, and select Export… from the context menu.

**Read more:** Copying a Chart or Table (page 734); Exporting a Chart (page 734).

---

**Note:** When viewing time history results from a SubModel within the parent model, the Chart display is modified somewhat, and in some cases provides slightly different options.

---

**Viewing a Time History Table**

When you display a Time History result by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element’s primary output) and selecting Time History Result… from the context menu., by default, a chart will be displayed.

**Read more:** Viewing a Time History Chart (page 584).

You can view a Time History Table by pressing the Display Table button when viewing a chart.

---

**Note:** When viewing a Time History Result element, the element “remembers” the last type of view that was displayed, and displays that view when you double-click on it.

---

A Time History Table looks like this:
For multi-realization runs, there are three header rows. The first shows the Label for the result; the second shows the unit; and the third identifies exactly what is being displayed (e.g., a realization or a specific statistic). For single realization runs, the third header row is not shown (as it is not applicable).

The first column then shows the plot points being used (i.e., scheduled timesteps at which point values are saved). Each row represents a separate plot point (a result value that was saved at a particular point).

What is displayed in the first column is a function of two things:

- Is the **Time Display Setting** (in the Properties dialog) set to “Simulation Time”, “SubModel Time” or “Reporting Periods”? In the first two cases (“Simulation Time is the default and the only option for interactive results), the first column contains elapsed times or dates. In the last case, the first column contains Reporting Period names.

  **Read more:** Viewing Reporting Period-Based Results in Time History Result Elements (page 607).

- Is the **Time Basis** in the Simulation Settings dialog “Elapsed Time” or “Calendar Time”? In the former case, the first column displays elapsed times. In the latter case, the first column displays dates.

  **Read more:** Defining the Time Basis and Simulation Duration (page 453).

You can increase or decrease the width of a column by dragging the line separating columns to the right or to the left, respectively. If the column is too narrow to display the contents of a cell, holding the cursor over the cell displays a tool-tip with the contents.

The Time History Table has a variety of buttons and controls at the top of the window. Some of items are always present, while others are only shown under certain circumstances (e.g., when viewing multiple realizations and/or multiple outputs).

The following three buttons are always available:

**Display Chart:** Selecting this button switches to a Time History Chart view of the result. Note that when viewing a table, the Display Table button
Viewing Time History Results

appears selected; when viewing a chart, the Display Chart button appears selected.

![Chart Button](chart_button.png)

**Edit Properties:** This provides access to the Result Properties.

![Edit Properties](edit_properties.png)

**Read more:** Viewing the Properties of a Time History Result (page 581).

**Sort:** This button allows you to sort the rows based on a selected column.

![Sort Button](sort_button.png)

**Read more:** Sorting Values in Result Tables (page 568).

Several additional points should be noted regarding Time History Tables:

- Conditions are displayed in tables as either 1/0, true/false, True/False, TRUE/FALSE, on/off, or High/Low. You select which of these pairs to use on the **Results** tab of the Options dialog (accessed via **Model | Options…** from the main menu).

- You can control the number of significant figures displayed in tables from the **Results** tab of the Options dialog (accessed via **Model | Options…** from the main menu).

**Read more:** Controlling Significant Figures and Scientific Notation in Result Displays (page 568).

- You can copy the contents of a Time History Table to the clipboard by pressing **Ctrl+C**. You must first select the cells you wish to copy. You can do so by placing your cursor in one cell and dragging to another location. You can select the entire table by pressing the cell in the upper left-hand corner of the table. Cells that are not adjacent can be selected by pressing the **Ctrl** key when selecting. Selected items will be highlighted in black. You can subsequently paste the table into another application (such as a spreadsheet).

**Read more:** Selecting Items and Copying Values in Result Tables (page 567).

**Note:** When viewing time history results from a SubModel within the parent model, the Table display is modified somewhat, and in some cases provides slightly different options.

**Read more:** Viewing SubModel Results in Time History Result Elements (page 617); Carrying Out Nested Monte Carlo Simulation Using a SubModel (page 992).

It is often useful to plot the time histories of multiple outputs on a single plot. This allows you to compare and contrast different results, or to better understand how various parameters in your model affect each other. Therefore, GoldSim allows you to display multiple outputs on a single time history chart or table.
To display multiple results in a Time History Result, you must first open the Result Properties dialog for the result.

**Read more:** Viewing the Properties of a Time History Result (page 581).

You can do this by pressing the Edit Properties... button in any of the Result display windows:

The following dialog will be displayed:

This is the dialog for an interactive result. A Result element will have a Name defined. Note that the Label is user-editable and is used in legends and column headers.

To add additional outputs to the result, press the Add Result... button. A dialog for selecting an output will be displayed. After you select the output you wish to add to the result display, and press OK, the selected outputs are then shown in the Result Properties dialog:
You can show the full path for each result (showing the containment hierarchy) by placing your cursor over the Result item to display a tool-tip.

Results can be deleted with the **Delete Result** button (holding the Ctrl key down changes this to a **Delete All** button). The order in which multiple results appear in the list can be changed using the **Move Up** and **Move Down** buttons.

The **Show Result** >> button opens a result display window (while keeping the Properties dialog simultaneously open).

**Read more:** Viewing a Time History Chart (page 584); Viewing a Time History Table (page 586).

Pressing the **Show Result** button displays the multiple output display.

A Time History Chart with multiple outputs looks like this:
The legend displays the (user-editable) Labels defined for each result.

Note that a legend is available for Time History Charts (you may have to turn it on by right-clicking in the chart and selecting to show the legend from the context menu). The Labels specified in the Result Properties dialog are used in the legend to label the different results.

For Time History Tables, the results are listed in separate columns:

![Time History Result](image)

The column headers are the (user-editable) Labels defined for each result.

The following points should be noted regarding the Time History Result Properties dialog when displaying multiple results:

- For each result in the list, you specify a Style (used in charts), and the Label used in legends in charts and headers in tables.
  
  Read more: [Controlling the Chart Style in Time History Results](page 635).

- The Y1 and Y2 checkboxes are used to determine which axis the result is plotted on. If you add a second output that has the same dimensions as the original output, it will automatically be placed on the primary (Y1) axis. If it has different dimensions, it will automatically be assigned to the secondary (Y2) axis. You can add as many outputs as desired to the list. If you add an output whose dimensions do not match those currently assigned to the Y1 or Y2 axes, it will be assigned to neither. As a result, it will not be displayed in the chart (but will be displayed in the table). If you specifically try to assign an unassigned output to an axis, any outputs assigned to that axis with different dimensions will be cleared and the axis will take on the new dimension.

- You can specify the Display Units for the Y1 and Y2 axis (which overrides the display units specified within the element’s property dialog).

- When you are viewing time histories of multiple realizations of multiple outputs, the display windows provide a number of options that allow you to select how you want to view the results. Some options allow you to view probabilistic results for one result at a time (e.g., all realizations for a single result), while others allow you to view probabilistic results for all results simultaneously (e.g., a particular statistic for each result).
Viewing Time History Results

**Viewing Time Histories for Array Outputs**

When you wish to view time history results for an array output, you must first choose whether you wish to view results for a single item of the array, for selected items, or for all items.

If you wish to view results for only a single item, you should expand the array in the browser or output interface, right-click on the specific item which you wish to view, and select **Time History Result**... from the context menu.

Alternatively, you could do the same thing in the Result Properties dialog (i.e., select a single item) when adding the output to the list.

**Note:** When you create a new array element, the **Time History** checkbox defaults off (and hence time histories are not saved). If you wish to save time histories for an array, you must specifically check this box (or reference the output in a Time History Result element).

**Read more:** [Viewing Time Histories of Multiple Realizations for Multiple Outputs](page 605).

If, however, you wish to view selected items (or all items) of the array, you should either right-click on the entire array, and select **Time History Result**... from the context menu, or add the entire array to the list in the Result Properties dialog. If you do so, the Result Properties dialog looks like this:

**Read more:** [Specifying Results to be Saved](page 494).
Note that in the **Result** column, the type of array (in this case, a vector) is specified.

In addition, the **Style** column includes an **Edit...** button for each array result. This button can be used to specify the Style for each item being plotted (e.g., line color), and can also be used to specify which items will be included in the display.

Pressing the **Edit...** button for a vector result displays an option to edit the Array Label set defining the vector:

In this example, selecting this option would display the “Fruit” Array Label Set:
This dialog allows you to change the **Style** for each item used for charts. In addition, the **Show** checkbox allows you to select which items are included in the displays (by default all will be checked).

If you choose to display a time history chart of an entire vector, the chart will look something like this.

In this example, the vector has four items (Apples, Oranges, Peaches and Pears).
GoldSim will display each item of the vector, and add a legend. The style of each line is defined by the **Style** in the Array Label Set dialog, and the legend lists the array item names.

**Note:** If you do not want to repeat the name of the array (Production in the example shown above), you can specify a blank Label for that result item in the Properties dialog.

The table view of a vector would look like this:

![Table View](example_table.png)

Each item of the vector is displayed in a different column.

Pressing the **Edit**... button for a matrix result displays an option to edit both the row and the column Array Label sets defining the matrix:

![Edit Option](example_edit.png)

This allows you to select specific rows or columns to include/exclude in the displays, and also allows you to define the **Style** for each item in charts.

When displaying charts and tables for a matrix, GoldSim, by default, displays all items. For example, the chart below displays a matrix with 4 rows and 2 columns:
In such a chart, how are the Styles determined, since each row and each column has a specified Style (defined in the Array Label Set dialog) and each item being plotted is associated with both a row and a column? When plotting a matrix, the following rules are applied for each item:

- The color for an item is based on the Style for the row; and
- All other attributes, such as the line width, style (e.g., solid, dashed), symbol style, symbol size and symbol color, are based on the Style for the column.

The table view of a matrix looks like this:

Each item of the matrix (which has been selected to be shown in the Array Label Set dialogs) is displayed in a different column of the table.

When you are viewing time histories of multiple realizations of an array, the display windows provide a number of options that allow you to select how you want to view the results. Some options allow you to view probabilistic results for one item at a time (e.g., all realizations for a single result), while others allow you to view probabilistic results for all items simultaneously (e.g., a particular statistic for each result).
When you are viewing time histories after you have run multiple realizations, there are a number of different ways that you can view the results (e.g., view one realization at a time, view all realizations, view a particular statistic). The Display windows provide a number of options that allow you to select how you want to view the results.

**Note:** Time History results for simulations with multiple realizations can only be displayed for outputs that are referenced by Time History Result elements.

**Note:** You can only view those realizations which are available (i.e., those which have been saved and have not been screened out).

If you have run multiple realizations, the top portion of the display window (either a chart or a table) for a Time History result looks like this:

As can be seen, the **Display drop-list** provides five options:

- **Realization**: This displays a single realization at a time. A control is added to the display window to select any (unscreened) realization:

  By default, the final (unscreened) realization is shown. Note that you can also toggle to other realizations using the **Ctrl+Up** and **Ctrl+Down** keys.

- **All Realizations**: This displays all (unscreened) realizations simultaneously. The chart will show multiple lines. The table displays a column for each realization.

  **Note:** If the Time History Result element contains multiple outputs and/or an array, the display window provides a field for selecting a single result to be displayed when this option is selected.

- **Probabilities**: This is the default display. It provides a **probability histories** display. This is a powerful probabilistic representation of the time history of an output. In particular, the multiple realizations of the time history are represented by plotting (or tabulating) the statistics (e.g., percentiles, bounds, mean) of the set of time histories:
Note: If the Time History Result element contains multiple outputs and/or an array, the display window provides a field for selecting a single result to be displayed when this option is selected.

- **Statistic**: This displays a single specified statistic for the collection of (unscreened) realizations. A control is added to the display window to specify the statistic:

  ![Statistic control](image)

  The default statistic that is shown is defined (and can be changed) in the Monte Carlo Result Display dialog (accessed via the Options… button in the Result Properties dialog).

  **Read more**: [Controlling Monte Carlo Result Options](page 482).

  Pressing the Percentile… button allows you to define a specific percentile:

  ![Percentile dialog](image)

  The Statistic option is of particular value when displaying multiple results, as it allows you to quickly view a single statistic for all results.
• **Custom Statistics**: This displays a different custom statistic for each result (as opposed to the same statistic for each result). The Custom Statistic that is shown is specified in the Result Properties dialog.

The Custom Statistic option is valuable if you want to display different statistics for different outputs, or several different statistics for the same output:

*Read more: [Viewing Custom Statistics for Multiple Realizations](page 603).*

---

**Note**: The first time that a Time History Result element is displayed, it will display Probabilities (the default). However, when viewing a Time History Result element, the element “remembers” the last type of display that was selected, and shows that when you double-click on it the next time.
Viewing Time History Results

Viewing Probability Histories for Multiple Realizations

If you have saved multiple realizations for one or more outputs (by referencing them in a time History Result element), by default, a probability histories display will be shown:

![Probability Histories Example](image)

**Note:** The first time that a Time History Result element is displayed, it will display Probabilities (the default). However, when viewing a Time History Result element, the element “remembers” the last type of display that was selected, and shows that when you double-click on it the next time.

A probability histories display provides a probabilistic representation of the time history of the output. In particular, the multiple realizations of the time history are represented by plotting (or tabulating) the percentiles (as well as the bounds and mean) of the set of time histories for the output.

By default, the areas between the percentile curves are filled. You can show just the lines by clicking the **Show Area or Line** button at the top of the chart:
A table looks like this:

Each statistic is displayed in a different column.

By default, GoldSim plots the following pairs of percentiles when displaying probability histories:

- Min/Max (0%/100%)
- 1%/99%
- 5%/95%
- 15%/85%
- 25%/75%
- 35%/65%
- 45%/55%

It also plots the median (50th percentile). The mean is off by default.
You can customize which of these statistics you would like to plot, as well as add some of your own. You can do so via the Monte Carlo Result Display dialog (accessed via the Options… button in the Result Properties dialog).

Read more: Controlling Monte Carlo Result Options (page 482).

The top of this dialog looks like this:

![Monte Carlo Result Display Properties](image)

You can activate or deactivate a particular line (or pair of lines) by clicking in the checkbox to the left of the item. You can add new percentile pairs by pressing the Add… button. The following dialog is displayed.

![New Percentile](image)

You need to only type in one item of a pair (as a percentage, not a fraction), and the complementary item will also be added. For example, if you enter 10, both the 10th and 90th percentiles will be added.

You can subsequently remove any percentile pairs using the Delete button.

Note: There must be a minimum of two statistics selected: the Mean or Median (50%), and at least one percentile pair.

Warning: You should only view particular percentiles if you have run a sufficient number of realizations such that that statistic can be accurately estimated. As a general rule of thumb, in order to view the Xth percentile (where X is the lower number of the pair, expressed as a fraction), at least 5/X realizations should be available. For example, in order to view the 10th (and 90th) percentile, at least 5/0.1 = 50 realizations should be available. GoldSim does not enforce this limitation automatically (it will try to display any percentiles that you select regardless of the number of realizations available). However, you should manually adjust the percentiles that you choose to view based on the number of realizations available.
Note: The percentiles are truncated at the actual least and greatest values for each timestep. In the absence of importance sampling, these will correspond to probability levels of 1/2N and 1 - 1/2N, respectively. For example, if you run 10 realizations, any percentile line below the 5th percentile will be coincident with the 5th percentile.

For each statistic, you select a Style for the line and fill color used for the chart. By default, the Auto-fill color range option is checked. This option makes it easy to quickly create color gradients for the Styles. Whenever you select one of the Styles (and change a color), GoldSim auto-generates colors for the other statistics to create a logical gradient (e.g., if you select red, all the other statistics will be changed to shades of red). If you want to select your own colors for each statistic, you must clear the Auto-fill color range option.

Note: Autofill does not apply to the Mean and the Median (50%). These can be edited separately without affecting the color gradient.

The following additional points should be noted regarding displaying probability histories:

- When you select Mean, GoldSim will also display the Standard Deviation (SD) in tables. However, the SD does not appear in charts.
- All of the information specified in the Monte Carlo Result Display dialog (i.e., the percentiles to display and the styles) is saved with the model and are applied to all probability history displays in the model.
- If the Time History Result element contains multiple outputs and/or an array, the display window provides a field for selecting a single result to be displayed when probability histories are selected.

Read more: Viewing Time Histories of Multiple Realizations for Multiple Outputs (page 605); Viewing Time Histories of Multiple Realizations for an Array (page 606).

If you have saved multiple realizations for one or more outputs (by referencing them in a time History Result element) one of the display options is to show Custom Statistics:

The Custom Statistic option is valuable if you want to display different statistics for different outputs, or several different statistics for the same output:
Note that the legend displays the result and the statistic being displayed.

**Note:** The first time that a Time History Result element is displayed, it will display Probabilities (the default). However, when viewing a Time History Result element, the element “remembers” the last type of display that was selected, and shows that when you double-click on it the next time.

The Custom Statistic that is shown for a particular result is specified in the Result Properties dialog:

There is a default custom statistic provided in the list:

The default statistic that is shown is defined (and can be changed) in the Monte Carlo Result Display dialog (accessed via the Options… button in the Result Properties dialog).

*Read more:* Controlling Monte Carlo Result Options (page 482).

Pressing the Percentile… button allows you to define a specific percentile:
The Custom Statistic option is valuable if you want to display different statistics for different outputs, or several different statistics for the same output.

If you choose to display a time history of multiple realizations for multiple outputs, what is shown is a function of what has been selected from the Display list:

In particular, if **Realization, Statistic** or **Custom Statistics** is selected, GoldSim shows the selected result for each output in the list:

This is possible because in these three cases, there is only one set of results to display for each output (e.g., the selected realization for each output).

However, if **All Realizations** or **Probabilities** is selected, only one output (result) can be viewed at a time, and it is necessary to select which output you wish to view, since these two displays require multiple sets of results for each output (e.g., all realizations for a selected output):
The result that is displayed is selected directly to the right of the Display list (after “for”). In the example above, the result named “Volume” has been selected.

If you choose to display a time history of multiple realizations for an array, what is shown is a function of what has been selected from the Display list:

In particular, if Realization, Statistic or Custom Statistics is selected, GoldSim shows the selected result for each item of the array:

In this example, the vector consists of 4 items (Apples, Peaches, Oranges and Pears).

This is possible because in these three cases, there is only one set of results to display for each array item (e.g., the selected realization for each item).
However, if **All Realizations** or **Probabilities** is selected, only one array item can be viewed at a time, and it is necessary to select which item you wish to view, since these two displays require multiple sets of results for each item (e.g., all realizations for a selected item):

The item that is displayed is selected directly to the right of the **Display** list (after “for”). In the example above, the result named “Production[Apples]” (i.e., the item named “Apples” from the output “Production”) has been selected.

In some cases, you need to display accumulated, average, the change or the rate of change of values over specified periods (e.g., monthly, annually). For example, you may need to report the cumulative amount of money or water that moved from one point to another each month.

To support this, GoldSim allows you to define Reporting Periods. Scheduled updates (timesteps) are created at the end of each Reporting Period. Hence, these can also be thought of as “Reporting Steps”. However, whereas results saved at Basic Steps always represent instantaneous values, results saved at Reporting Periods can not only be displayed as instantaneous results, but can also report accumulated, averaged, the change or the rate of change of values over a specified period (e.g., monthly).

**Read more:** [Defining Reporting Periods](#) (page 461).

Results based on Reporting Periods (e.g., cumulative values over each period, average values over each period, etc.) can be accessed and viewed via Time History Result elements.

It is important to understand that the act of creating Reporting Periods does not in itself provide specialized Reporting Period-based results (e.g., accumulated or averaged results). This is because Reporting Periods by default simply provide another way to define timesteps, since as noted above, scheduled updates (timesteps) are created at the end of each Reporting Period. By default, Time History results simply provide instantaneous values at each timestep (and hence, if Reporting Periods are defined, at the end of each period).

To display specialized Reporting Period-based results, you must add the outputs you are interested in to a Time History Result element, and then specify that you want to view Reporting Period-based results.

By default, the dialog for a Time History Result element looks like this:
Note: For multiple realization runs, the Display column is replaced with Custom Statistic.

Read more: Viewing Time Histories of Multiple Realizations (page 597).

In particular, note that by default, the Time Display Setting in this dialog is set to “Simulation Time”. In this case, Time History results simply provide instantaneous values at each timestep (and hence, if Reporting Periods are defined, at the end of each period).

If you have defined Reporting Periods, you can select “Reporting Periods” for the Time Display Setting. When you do so, the Result Properties dialog looks like this:
Note that a new column has been added, titled **Period Results**. This is a drop-list that allows you to select results based on Reporting Periods. The list consists the following options:

- **All plot points**: This displays instantaneous values at all the plot points (scheduled updates). This would include not only values at the end of each Reporting Period, but also values at all other scheduled updates (e.g., Basic Steps). This display is identical to what you would see if you selected “Simulation Time” for the **Time Display Settings**. Note, however, that the values of all plot points only appear in charts. In tables, only values at the end of each Reporting Period are shown.

- **At period end**: This displays instantaneous values at the end of each Reporting Period. It is the default.

- **Average**: This computes the average value for each Reporting Period. For example, if you had a Basic Step of 1 day, a monthly minor Reporting Period and annual major Reporting Period, GoldSim would display either monthly or annual averages (you select in the display window which to show).

- **Change**: This computes the change in the value over each Reporting Period (the difference between the value at the end of the current Reporting Period and the value at the end of the previous Reporting Period).

- **Rate of Change**: This computes the rate of change in the value over each Reporting Period (from the end of the previous Reporting Period to the end of the current Reporting Period). That is, it is the Change (defined above)
divided by the length of the Period. It has dimensions of $X/\text{Time}$, where $X$ is the dimensions of the actual output.

- **Cumulative**: This computes the integral of the value over each Reporting Period (from the end of the previous Reporting Period to the end of the current Reporting Period). It has dimensions of $X\times\text{Time}$, where $X$ is the dimensions of the actual output. This would typically be used in cases where the output was a rate (e.g., $\text{m}^3/\text{day}$), such that the cumulative value represents the quantity (e.g., $\text{m}^3$) that flowed or accumulated over the period.

**Note**: Most spreadsheet models implicitly compute *accumulated* values (e.g., cumulative flows) over a step. Hence, if you want to compare GoldSim with a spreadsheet model, you should use Reporting Periods to define your steps, and view the Cumulative result.

**Note**: If your result is a condition, only two of the Period Result types are applicable: “All plot points” and “At period end”. Hence, these are the only two choices in the drop-list.

**Note**: In order to display most types of Reporting Period results, you must add the result to the Result Properties dialog before you run the model (i.e., while you are in Edit Mode). This is because they are actually computed “on the fly” as the model is running (e.g., the Cumulative option integrates values during the simulation). Hence, if you try to add a result to a Time History Result element with a **Time Display Setting** of “Reporting Periods” while in Result Mode, the only available Reporting Period options are “All plot points” and “At period end”.

To understand the use of Reporting Period-based results, it is instructive to view an example. In the example below, the same output (Flow_Rate) is added to the Result element twice:
In this particular model, a daily Basic Step has been defined, along with a Major (annual) and minor (monthly) Reporting Period. The first item in the list instructs GoldSim to display the instantaneous value of the output at all plot points. The second item in the list instructs GoldSim to display the average value of the output over the Reporting Period.

If you have defined both a Major Period and a Minor Period, you must specify which Period to display when viewing results. In this example, you would need to select whether you wish to display monthly or annual averages at the top of the display. In this case the monthly values are being shown:

In this case the annual values are being shown:
Note that the table only displays results corresponding to the actual reporting periods (even though the first item, labeled “Daily Flow” here, was selected to display “All plot points” in the Properties dialog). Hence, in this table, the column labeled “Daily Flow” displays the instantaneous values at the end of each period, while the column labeled “Average” displays the average values over each period. As will be shown below, however, when “All plot points” is selected for an item, all points are displayed in a chart.

When displaying charts of Reporting Period-based results, you must specify how you would like the values to be plotted. This is because, for four of the result types (Average, Change, Rate of Change and Cumulative), the results actually apply for an entire period (as opposed to the end of the period). To represent this, GoldSim provides two options for plotting these results, illustrated in the examples below. In these examples, the Daily Flow and the Average are displayed on the same chart:

In this case, the major (annual) Reporting Period Average is plotted as a horizontal line over the period (a stair-step). Here is the same plot displaying the minor (monthly) Reporting Period results:
Alternatively, the plot can be displayed like this (these are showing annual results again):

In this case, the major (annual) Reporting Period Average is plotted as a line connecting the **mid-point** of each period (this can be seen better by changing the style to add a symbol at each data point):
Here is the same plot (without the symbols) displaying the minor (monthly) Reporting Period results with lines drawn between mid-points:

The type of plot is controlled by this button at the top of the display window:

If the button is pressed, the plot is shown as a stair-step; otherwise it is shown as lines connected the mid-points.

Note that this option (to plot as a stair-step) is only used when you have selected to display Average, Change, Rate of Change or Cumulative Reporting Period-based results. If you select “All plot points”, all points are plotted (and connected by lines), as shown in the “Daily Flow” line above. If you select “At period end”, the values are plotted at the end of each period (as opposed to the mid-points) plotted (connected by lines):
Note that if you select either “Cumulative” or “Rate of change”, and you simultaneously select any of the other result types, the Y2 axis must also be used, since the dimensions of the result are different from that of the output:

A simple example file illustrating Reporting Periods (ReportingPeriods.gsm) can be found in the General Examples/ Running folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

When carrying out probabilistic simulations, you may often run hundreds or thousands of realizations. In order to analyze the results, it is often quite useful to classify the realizations into categories. A category is simply defined by a condition relating one or more outputs in the model (e.g., those realizations in which the discount rate was above 3.5%; those realizations in which the profit exceeded $1,000,000; those realizations in which the peak concentration was between 1 mg/l and 10 mg/l).

Read more: Classifying and Screening Realizations (page 577).
Classification categories are defined at the bottom of the Monte Carlo Result Display Properties dialog:

![Monte Carlo Result Classification and Screening](image)

This dialog can be accessed from the Monte Carlo tab of the Simulation Settings dialog (via the Result Options… button), and is also accessible by pressing the Options… button available in the Result Properties dialog of a Time History Result element.

When viewing a Time History result, if 1) you have run multiple realizations; 2) you have defined more than one category; and 3) you are displaying All Realizations, GoldSim will label the curves in the Time History Chart based on the categories you have defined:

![Time History Chart with Category Labels](image)

Read more: Viewing Time Histories of Multiple Realizations (page 597).

In the example shown above, three categories have been defined. The chart is displaying all realizations for a single result, with each curve having a style defined by its category. (Note that for each category, you can edit the Style via the Monte Carlo Result Display Properties dialog.)

In addition, screening by category is applied when displaying time histories. That is, if you have chosen to screen out one or more categories (by clearing the Include box in the dialog above), the results (e.g., statistics, realizations) that are shown (in charts and tables) only include those realizations in the categories which you have chosen to include. In the example below, realizations falling into the Medium and High categories have been screened:
Note that the legend indicates that some categories have been screened out.

**Viewing SubModel Results in Time History Result Elements**

In most cases, when you add an output to a Time History Result (via the Add Result... button in the Result Properties dialog), you will simply be adding a standard output (such as the output of an Expression element or the primary output of a Reservoir element). However, in one special case, you can add (and view) a specialized output referred to as a Time History Definition output in a Time History Result.

Time History Definition outputs are complex outputs that represent all the information necessary to define a time series. These outputs can be identified when viewing the output port of an element:

In this example, "Volume_in_Pond" is a Time Series Definition Output. Note that the Time History Definition output is highlighted with a gray icon.

Time Series Definition outputs can be produced by several types of elements, but only those produced by SubModels can be added to a Time History Result.

Read more: Using SubModels to Embed Models Within Models (page 973); Creating the Output Interface to a SubModel (page 984).

Adding a Time History Definition output of a SubModel to a Time History Result in the parent model provides a mechanism to view the time history results generated by the SubModel directly within the parent model.
In order for SubModel time history results to be displayed in the parent model, you must do the following:

1. Create a Time History Result element in the parent model before running the model (i.e., while the model is in Edit Mode).
2. Within the Result Properties dialog, select “SubModel Time” for the Time Display Setting:

![Time History Result Properties](image)

3. When you do so, this restricts the types of results that can be added via the Add Result… button. In particular, when you press this button, you will only be presented with a list of SubModels, and you can only add SubModel Time History Definition outputs:

![Select the output to link to](image)

Several points should be noted:
• You can only add results from a single SubModel. Hence, once you add results from one SubModel, any subsequent results that you add must come from the same SubModel. If you add results from different SubModels, you will not be able to run the model.

• There are three different types of Time History Definition outputs that can be generated by a SubModel (Last Calculated, Statistic History, Realization Histories).

  **Read more:** Creating the Output Interface to a SubModel (page 984).

  If you want to plot Realization Histories, it is the only result that can be added to the Result element. If you add multiple results, and one of the results consists of Realizations Histories, you will not be able to run the model.

• The time history results associated with a SubModel can (and typically will) have completely different Time settings than the parent model. When viewing time history results associated with a SubModel in the parent model, these are always based on the Time settings within the SubModel.

• Similarly, the results from a SubModel can (and typically will) have completely different Monte Carlo settings than the parent model. When viewing time history results associated with a SubModel in the parent model, probabilistic results will appropriately reflect the Monte Carlo settings of both the SubModel and the parent model.

• You cannot export time history results (i.e., to a spreadsheet or text file) from a SubModel (i.e., if the Time Display Setting in the Result element is “SubModel Time”).

• Scenario results are not saved for time history results associated with a SubModel. If a Result element in a parent model is linked to SubModel Time History Definition outputs, no results will be displayed in Scenario Mode.

  **Read more:** Viewing Scenario Results in Time History Result Elements (page 622).

**Note:** The Time settings of the parent model have no effect on how time history results associated with a SubModel are plotted in a Time History Result element within the parent model.

As pointed out above, when creating a Time History Definition output for a SubModel, three types of outputs can be specified:

  **Last Calculated:** This represents a single time history of the output for the final realization of the SubModel (which is equivalent to the only realization if the SubModel Monte Carlo Settings do not specify multiple realizations).

  **Statistic:** This represents a single time history for a specified statistic (computed over all realizations of the SubModel).

  **Realization Histories.** This is the time history of the output for all realizations of the SubModel.

  **Read more:** Creating the Output Interface to a SubModel (page 984).
The options provided for displaying time history results associated with a SubModel in a Time History Result element within the parent model are a function of the type of Time History Definition output being displayed and the Monte Carlo settings of the SubModel and parent model.

### Viewing Last Calculated and Statistic History Results from a SubModel

Due to the manner in which they are defined, “Last Calculated” and “Statistic” Time History Definition outputs produce a single time history for each realization of the parent model (regardless of how many realizations were carried out for the SubModel).

Therefore, the manner in which these results are viewed is identical to how time histories of multiple realizations would be viewed for any other type of output. In particular, when you are viewing time histories of Last Calculated or Statistics for a SubModel output after you have run multiple realizations of the parent model, there are a number of different ways that you can view the results (e.g., view one realization at a time, view all realizations, view a particular statistic). The Display windows provide a number of options that allow you to select how you want to view the results:

[Display options image]

**Read more:** [Viewing Time Histories of Multiple Realizations](page 597).

---

**Note:** If only one realization of the parent model is run, only one option (Display Realization or Display History) will be available.

### Viewing Realization Histories Results from a SubModel

Viewing “Realization Histories” outputs from a SubModel is considerably more complex than viewing “Last Calculated” and “Statistic” outputs, since “Realization Histories” Time History Definition outputs can produce *multiple* time histories for each realization of the parent model. This is because “Realization Histories” consist of time histories of all realizations of the SubModel. For example, if the parent model was run for 100 realizations, and the SubModel was run for 50 realizations, the results being displayed in a Time History Result element within the parent model would be based on 5000 individual time histories (50 * 100).

Fortunately, GoldSim provides some powerful tools to view this complex set of results. To better understand these tools, it is instructive to first understand how such results would be viewed in a simpler case: a model in which the SubModel was run for multiple realizations, but the parent model was run only for a single realization. In such a case, the Result Display (for a chart) would look like this:
In this particular case, the manner in which these results are viewed is identical to how time histories of multiple realizations would be viewed for any other type of output. The Display windows provide a number of options that allow you to select how you want to view the results.

Read more: Viewing Time Histories of Multiple Realizations (page 597).

Note: If only one realization of the SubModel is run, only one option (Display History) will be available.

However, there is one very important distinction that is critical to note. This result is not displaying multiple realizations of the parent model (recall that only a single parent model realization was carried out). This result is displaying multiple realizations from within the SubModel. That is, although the Result element is located in the parent model, in this particular case, it is displaying the realizations carried out by the SubModel. This is clearly indicated in the Display drop-list by listing "(SubModel)" after each of the five display options. This is also indicated when displaying tables in the manner that the histories are labeled:
Note that that the various time histories are labeled as H#1, H#2, and so on. For realizations of the parent model, they would be labeled as R#1, R#2, etc. The idea is that H refers to a SubModel “history”, while R refers to a parent model “realization”.

Result display is more complex when both the parent model and the SubModel are run for multiple realizations (referred to as “nested Monte Carlo simulation”):

As can be seen, multiple options are provided to allow you to view the nested Monte Carlo results. These options are discussed in detail in the section discussing nested Monte Carlo simulation listed below.

Read more: Carrying Out Nested Monte Carlo Simulation Using a SubModel (page 992).

GoldSim’s scenario modeling capability allows you to directly compare results generated by different sets of input parameters. In effect, when you use this capability, your model can store (and subsequently compare) multiple sets of results (and inputs).

Read more: Creating, Running and Comparing Scenarios (page 504).

When a model is in Scenario Mode, Time History Result elements can be used to view scenario results.

In order for scenario results to be displayed in a Time History Result element, the Show button must be checked for each scenario to be displayed in the Scenario Manager (accessed from the main menu (Run | Scenario Manager…)) or by pressing F7):
In this example, the Show box is checked for all three scenarios, so all three will be displayed in results.

If you double-click on a Time History Result element in Scenario Mode, GoldSim displays results for all scenarios for which scenario results have been generated (and for which the Show button has been checked from within the Scenario Manager).

Note that for each scenario, within the Scenario Manager you can edit the Style, as well as the Scenario Name. These affect how the results are labeled in chart and table displays.

Read more: Controlling the Chart Style in Time History Results (page 635).

The time history display for scenarios differs depending on how the simulation was run (single realization or multiple realizations), what you choose to display (e.g., a statistic or all realizations), and whether the Time History Result element contains multiple results.

In all cases, you can select which results and which scenarios you wish to display. For example, if only a single realization was run (for three different scenarios), the Table display in Scenario Mode would look like this:

Read more: Viewing a Time History Table (page 586).
By default, all scenarios are displayed for the first result listed in the Properties dialog for the Result element. Each column then represents a scenario. The result being displayed is indicated in the upper left-hand corner of the table.

The corresponding chart in Scenario Mode would look like this:

![Chart showing time history results with legend for volume over time for Low Leakage, Base Case, and High Leakage scenarios.]

**Read more:** Viewing a Time History Chart (page 584).

The Scenario Names are shown in the legend. The Result being displayed is indicated in the legend title.

You can specify exactly which scenarios and results you wish to display using a drop-list at the top of the display window. In the example below, there are two results and three scenarios:

![Dropdown menu showing options to select scenarios or results for time history chart.]

Selecting an option prefaced with “Result” displays all the scenarios for that particular result (as shown in the table and chart shown above). Selecting an option prefaced with “Scenario” displays all the results for that particular scenario (as illustrated below):
In this example, two different results are shown (Volume and Peak) for the selected scenario (High Leakage). The corresponding Table display would look like this:

<table>
<thead>
<tr>
<th>Day</th>
<th>Volume</th>
<th>Peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.57</td>
<td>9.57</td>
</tr>
<tr>
<td>2</td>
<td>14.4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>17.1</td>
<td>19.1</td>
</tr>
<tr>
<td>4</td>
<td>18.4</td>
<td>18.4</td>
</tr>
<tr>
<td>5</td>
<td>18.9</td>
<td>18.9</td>
</tr>
<tr>
<td>6</td>
<td>21.6</td>
<td>21.6</td>
</tr>
<tr>
<td>7</td>
<td>22.2</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>23.1</td>
<td>23.1</td>
</tr>
<tr>
<td>9</td>
<td>23.9</td>
<td>23.9</td>
</tr>
<tr>
<td>10</td>
<td>23.9</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>23.7</td>
<td>23.9</td>
</tr>
<tr>
<td>12</td>
<td>23.6</td>
<td>23.9</td>
</tr>
<tr>
<td>13</td>
<td>24.8</td>
<td>24.8</td>
</tr>
<tr>
<td>14</td>
<td>24.8</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>22.8</td>
<td>24.8</td>
</tr>
<tr>
<td>16</td>
<td>23.3</td>
<td>24.8</td>
</tr>
<tr>
<td>17</td>
<td>24.5</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>24.3</td>
<td></td>
</tr>
</tbody>
</table>

Each column represents a result. The scenario being displayed is indicated in the upper left-hand corner of the table.

There is also an option to display “All Results & Scenarios”: 
**Note**: Array results cannot be compared in Scenario Mode. Only scalar results can be displayed. Hence, if an array is listed as one of the results in the Result Properties page, it will not be available for selection in the display dialogs when comparing scenarios (i.e., when the model is in Scenario Mode). If you wanted to compare one or more items from an array in Scenario Mode, you would need to add those scalar array items as separate results in the Result Properties dialog.

**Note**: Scenario results are only available for results that are added to the Result element before you run a scenario. If you add a result after you run a scenario, scenario results for that output will not be available for that scenario.

If you have run multiple realizations for the various scenarios, what is shown is a function of what has been selected from the Display list:

In particular, if **Realization**, **Statistic** or **Custom Statistics** is selected, GoldSim shows the selected result for each scenario:
This is possible because in these three cases, there is only one set of results to display for each scenario (e.g., the selected statistic for each output).

However, if All Realizations or Probabilities is selected, only one scenario can be viewed at a time, and it is necessary to select which scenario (and output if multiple outputs are referenced by the Result element) you wish to view, since these two displays require multiple sets of results for each scenario (e.g., all realizations for a selected scenario and selected output):

The result and scenario that is displayed is selected directly to the right of the Display list (after “for”). In the example above, the result named “Volume” for scenario “Low Leakage” has been selected.

When viewing a Time History result in Scenario Mode, if 1) you have defined more than one category (defined at the bottom of the Monte Carlo Result Display Properties dialog); and 2) you are displaying All Realizations for a multiple realization run, GoldSim will label the curves in the Time History Chart based on the categories you have defined. For example, if your categories...
were defined as “Normal”, “Too Low” and “Danger”, a chart with categories might look like this:

Note that the chart is displaying a single result for a single scenario.

In addition, screening by category is active in Scenario Mode. That is, if you have chosen to screen out one or more categories (by clearing the Include box in the dialog above), the scenario results that are shown (in charts and tables) only include those realizations in the categories which you have chosen to include.

Note that depending on how you have defined the categories, it is possible for a realization to fall into a different category in different scenarios. Hence, when screening results in Scenario Mode, the number of realizations within each scenario may differ. In the example below (displayed as a table), realization #18 is included in the first two scenarios, but is screened from the third:

Read more: Using Result Classification and Screening in Time History Results (page 615).
Note: High resolution results (unscheduled updates) are never displayed in Scenario Mode. They are only available in Result Mode. Hence, you cannot view high resolution results when comparing scenario results. When comparing scenarios, only scheduled updates are included in displays.

Read more: Viewing Unscheduled Updates in Time History Result Elements (page 629).

Note: Scenario results are not saved for time history results associated with a SubModel. If a Result element in a parent model is linked to SubModel time history results, no results will be displayed in Scenario Mode.

Read more: Viewing SubModel Results in Time History Result Elements (page 617).

In some cases, events or other changes in the model may not fall exactly on a scheduled update. That is, some events or changes may actually occur between scheduled updates of the model. These trigger an “unscheduled update” of the model. Unscheduled updates are timesteps that are dynamically inserted by GoldSim during the simulation in order to more accurately simulate the system.

Read more: Understanding Timestepping in GoldSim (page 455).

Unlike scheduled updates, unscheduled updates do not normally appear in time history plots and tables. That is, although these timesteps may affect the results (e.g., by making them more accurate at the scheduled timesteps), unscheduled updates of the model are not saved and displayed. Only the scheduled updates are actually saved and displayed.

In some cases, however, it may be of interest to see the values of selected outputs at unscheduled updates. To facilitate this, GoldSim provides an option in the Advanced Time settings to save results at unscheduled updates (referred to as “high resolution” results). This option appears at the bottom of the Advanced Time Settings dialog (accessed from the Advanced… button on the Time tab of the Simulation Settings dialog):

By default, the checkbox labeled Include unscheduled updates (high resolution results) in scalar time histories is cleared. If you check this box, GoldSim will save unscheduled updates for time history results (referred to as “high resolution” results) under the following conditions:

- High resolution results can only be viewed from within Time History Result elements. If the output is not referenced by a Time History Result element, high resolution results will not be displayed.
- High resolution results are only available for single realization runs (Deterministic simulations or probabilistic simulation with a single realization).
- High resolution results are only available for scalar outputs.
- The Time Display Setting in the Result Properties dialog for the Time History Result element must be set to “Simulation Time”. High
resolution results are not available when viewing Reporting Period-based results.

- High resolution results are only available for results that are added to the Result element before you run the model. If you add a result after you run the model, it will only show results at scheduled updates.

If these conditions are met, and you view a Time History Result element, the Result Properties will look like this:

If all necessary conditions are met, the **Display** column will list “History (high res)”, indicating that the Result element will display high resolution results for that output.

In such a case, the display will include unscheduled updates. In the example below (showing a table display), scheduled updates occur every 10 days (there is a Basic Step of 10 days), but unscheduled updates occur at 2.46 days and 41.29 days:
Note that when saving “high resolution” results, to reduce storage requirements, GoldSim does not actually save every unscheduled and scheduled update. Rather, the GoldSim only saves results 1) at each update for which the result is different from the previous update; and 2) the previous update (i.e., the update immediately before the update where the value changed). This information is sufficient to create an accurate chart display. All other updates (representing updates during which the value did not change from the previous value) do not need to be saved.

In the example above, the value is changing continuously. Hence, all scheduled and unscheduled updates must be saved. In the example below, however, the value only changes at unscheduled updates (and remains unchanged in between), and hence there is no reason to save all scheduled updates; only those immediately preceding a change (at an unscheduled update) are saved:

The chart corresponding to the table shown above would look like this (symbols are placed at locations where a plot point occurs):

Note that there are no plot points for periods over which the plotted line is horizontal (since the value has not changed) even though there may have been scheduled (and unscheduled) updates during that period that did not affect the value.
The chart below displays the results for the two outputs in the example above in the same chart:

Note that Pond2 actually has more plot points than Pond1 (since it is changing continuously).

In some cases, it may be of value to view the same result with and without unscheduled updates. You can do this by taking advantage of the fact that high resolution results are only available for results that are added to the Result element before you run the model. If you add a result after you run the model, it will only show results at scheduled updates. Hence, to do such a comparison, you would 1) check Include unscheduled updates (high resolution results) in scalar time histories in the Advanced Time Settings dialog; 2) add a result to a Result element; 3) run the model; and 4) add the same result again after running the model. When you do this, the Result element will look like this:
Noe that the first item (added before the simulation was run) will display high resolution results, while the second item (added after the simulation was run) will display a normal time history.

In this example, scheduled updates occur every 10 days (there is a Basic Step of 10 days), and unscheduled updates occur at 2.46 days and 41.29 days:

If your Time History Result element included one scalar output and one array, the scalar would provide high resolution results, while the array would not. This would be indicated in the Result Properties dialog:
The Table display would show blank cells for the array at unscheduled updates:

Note that in this case, the high resolution results include all scheduled updates (since a row is provided for each scheduled update point due to the presence in the table of the array).

**Note:** High resolution results (unscheduled updates) are never displayed in Scenario Mode. They are only available in Result Mode. Hence, you cannot view high resolution results when comparing scenario results. When comparing scenarios, only scheduled updates are included in displays.

**Read more:** Viewing Scenario Results in Time History Result Elements (page 622).

A simple example file illustrating display of unscheduled updates (UnscheduledTimeSteps.gsm) can be found in the can be found in the General
Viewing Time History Results

Examples/ Running folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

Related Topics...

If an output is linked to a Time History Result element, GoldSim automatically saves the results for the element (even if you have manually specified in the element's property dialog or the property dialog for a parent Container that time histories are not to be saved).

Read more: Saving Outputs as Results (page 494); Controlling Result Flags for Elements in the Container (page 141).

Under some circumstances, you may want to temporarily disable saving the results for an element that is linked to a Time History Result element (e.g., if you wish to run a large number of realizations).

You can do this by checking the Disable Element checkbox at the bottom of the Time History Result element dialog:

Note that you can enable or disable all of the Time History Result elements within a Container directly from the Container's property dialog.

Read more: Controlling Result Flags for Elements in the Container (page 141).

GoldSim has powerful charting capabilities that allow you to customize the appearance of each chart. This includes adding headers and footers, and changing axis scales and labels. Most of these attributes can be edited by pressing the Edit Chart Style button at the top of the Time History Chart window:
Pressing this button (or right-clicking in a chart and selecting **Edit Chart Style...**) provides access to the following dialog for editing the various chart properties:

![Chart Style dialog](image)

This dialog is common to all types of charts, and is discussed elsewhere.

**Read more:** [Editing the Appearance of a Chart](#) (page 702).

One key attribute for charts is how values on axes are displayed. You cannot control the number of significant figures displayed (this is automatically determined). You can, however, control under what circumstances scientific notation is used via the **Results** tab of the Options dialog (accessed via **Model | Options...** from the main menu).

**Read more:** [Controlling Significant Figures and Scientific Notation in Result Displays](#) (page 568).

In addition to the basic chart attributes controlled by the Chart Style dialog, the Time History Result Properties dialog itself is used to control some of the attributes of a chart:
In particular,

- The units for each axis default to the Display Units of the first result added to each axis. However, you can change the **Display Units** that are displayed from within the Result Properties dialog.

- Only those results in which either the **Y1** or **Y2** box is checked will be included in displays. Hence, after adding a result to the list, you can temporarily hide it from displays by clearing one of these boxes.

- The **Label** is user-editable, and is used in legends (and column headers for tables).

**Note:** You can hide or show the legend on a chart via the context menu (i.e., accessed by right-clicking in the chart).

- The Style of each line displayed in a Time History Chart can be edited by clicking on the field (in the **Style** column) corresponding to each result. This provides access to the following dialog for editing the line style (for a distribution) or the fill style (for a single realization display):
Viewing Time History Results

Read more: Editing Data Styles (page 710).

If you are plotting an array, the Data Style for the different array items is defined in the Array Label Set dialog.

Read more: Viewing Time Histories for Array Outputs (page 592).

If you have run multiple realizations and you are displaying Probabilities, the Data Style for the display is specified at the top of the Monte Carlo Result Display Properties dialog (in the section labeled “History Statistics”). This dialog is accessed by pressing the Options… button in the Result Properties dialog of a Time History Result element.

Read more: Viewing Probability Histories for Multiple Realizations (page 600).

When viewing a Time History result, if 1) you have run multiple realizations; 2) you have defined more than one category; and 3) you are displaying All Realizations, GoldSim will label the curves in the Time History Chart based on the categories you have defined.

Read more: Using Result Classification and Screening in Time History Results (page 615).

The chart will display all realizations for a single result, with each curve having a style defined by its category. For each category, you can edit the Style via the Monte Carlo Result Display Properties dialog (accessed by pressing the Options… button in the Result Properties dialog of a Time History Result element):

Read more: Editing Data Styles (page 710).

The Style used for each category displayed in a Time History Chart can be edited by clicking on the field (in the Style column) corresponding to each category.
When a model is in Scenario Mode, Time History Result elements can be used to view scenario results. In this case, the line style for time history results for the different scenarios is controlled by the Scenario Manager dialog (accessed from the main menu (Run | Scenario Manager…)) or by pressing F7):

Read more: Viewing Scenario Results in Time History Result Elements (page 622).

In order for scenario results to be displayed in a Time History Result element, the Show button must be checked for each scenario to be displayed in the Scenario Manager.

Note that for each scenario, you can edit the Style, as well as the Scenario Name, which is used in legends (and column headers for tables).

**Viewing Distribution Results**

Distribution results provide a way to view the final values of probabilistic outputs. Distribution results can only be viewed for scalar outputs (or scalar items of arrays).

A Distribution result has three types of views:

- Distribution Summary;
- Chart View; and
- Table View.

If you have saved Final Values for an output, you can display a Distribution result by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element's primary output) and selecting Distribution Result… from the context menu. By default, a Distribution Summary will be displayed:
Note: When viewing a Distribution Result element, the element “remembers” the last type of view that was displayed, and displays that view when you double-click on it.

Read more: Creating and Using Result Elements (page 570).

If you click on the Result Properties button when viewing a Distribution Result chart or table, the Properties dialog for the result will be displayed. The Result Properties button is the furthest button to the right at the top of the display window:

Note that when you press this button, it does not close the result display window you were displaying. Both windows are displayed simultaneously (although you can subsequently close one or the other). When viewing results, any changes you make to the Result Properties are immediately represented in the result display.

The Result Properties dialog is always modal. That is, it cannot be minimized, and with the exception of viewing the result display (which can be displayed simultaneously and share the focus with the Properties dialog), it retains the focus while it is displayed. As a result, you must close the Result Properties dialog before you can edit any other part of your GoldSim model.

The Result Properties dialog for an interactive Distribution result looks like this:
At the top of the page is a button to **Create Element**. By pressing this button, a Result element is created. When you do so, the key change that will be noticeable is that the **Name** will now be editable. The **Create Element** button becomes an **Appearance** button (which is used to modify the appearance of the element itself).

*Read more:* Creating and Using Result Elements (page 570).

The **Show Result >>** button opens a result display window (while keeping the Properties dialog simultaneously open).

*Read more:* Viewing a Distribution Summary (page 643); Viewing a Distribution Chart (page 647); Viewing a Distribution Table (page 651).

The **Add Result…** button allows you to add other results to the chart. Results can be deleted with the **Delete Result** button. In addition to adding standard outputs (such as Expressions or the primary output of a Reservoir element), you can also add Distribution outputs, which are complex outputs that represent all the statistical information necessary to define a probability distribution (and can only be produced by certain elements).

*Read more:* Adding a Distribution Output to a Distribution Result (page 663).

The order in which multiple results appear in the list can be changed using the **Move Up** and **Move Down** buttons.

*Read more:* Viewing Distributions of Multiple Outputs (page 654).

For each result in the list, you specify a **Style** (used in charts), whether or not the result is to be included when displaying results (**Show**), and the **Label** used in legends in charts and headers in tables.

*Read more:* Controlling the Chart Style in Distribution Results (page 668).

You can also specify the **Display Units** for the X-axis of the result (which overrides the display units specified within the element’s property dialog).

If you have created categories (and have specified only a single result in the list), you will note that the various categories will also be listed in the dialog:
Categories allow you to classify the realizations into various groups in order to analyze the results.

Read more: Using Result Classification and Screening in Distribution Results (page 658).

The Properties dialog for a Distribution Result element has several differences:

- The Create Element button is replaced by an Appearance button (which is used to modify the appearance of the element itself).
- The Name (and Description) are editable (since this is an element).
- An Options… button is available that provides access to the Monte Carlo Result Display Properties.

Read more: Controlling Monte Carlo Result Options (page 482).
Viewing a Distribution Summary

If you have saved Final Values for an output, you can display a Distribution Result by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element's primary output) and selecting Distribution Result… from the context menu. By default, a Distribution Summary will be displayed.

If you are viewing a different type of display (either a chart or table), you can view a Distribution Summary by pressing the Chart or Table button at the top of the display (i.e., making sure both are buttons are cleared).

**Note:** When viewing a Distribution Result element, the element “remembers” the last type of view that was displayed, and displays that view when you double-click on it.

The Distribution Summary view of a distribution result is similar in appearance to the dialog used for defining a Stochastic element.

*Read more:* [Specifying the Distribution for a Stochastic Element](#) (page 155).

A Distribution Summary looks like this:

The *preview pane* (in the upper right-hand corner of the screen) shows a preview of the chart view of the distribution. You can copy the preview pane to the clipboard using Ctrl+C.

The left side of the screen displays a common set of percentiles for the distribution (in terms of Cumulative Probability/Value pairs).

The statistics for the distribution (*Mean*, 5% and 95% Confidence Bounds on the Mean, *Standard Deviation*, *Skewness*, and *Kurtosis*) are shown directly below the distribution's percentiles. The significance of these statistics is described in Appendix A. The number of realizations (*Number of Samples*) is also indicated.

The Calculator section of the window allows you to compute the value associated with a particular percentile or the percentile associated with a particular value:
Enter the percentile in the Cumulative Probability field and the corresponding value will be displayed in Value OR Enter the value in the Value field and the corresponding percentile will be displayed in Cumulative Probability.

The Calculator also displays the Probability Density and the Conditional Tail Expectation for the specified Cumulative Probability/Value pair. The Conditional Tail Expectation is the expected value of the output given that it lies above a specified Cumulative Probability. That is, it represents the mean of the worst 100(1 - α)% of outcomes, where α is the specified Cumulative Probability.

**Note:** Calculation of the Conditional Tail Expectation is discussed in detail in Appendix B.

The Distribution Summary has a variety of buttons at the top of the window:

**Display PDF:** This displays the preview pane as a probability density function (PDF):

**Display CDF:** This displays the preview pane as a cumulative distribution function (CDF):

**Note:** The algorithm used to create a PDF plot is discussed in detail in Appendix B.
Display CCDF: This displays the preview pane as a complementary cumulative distribution function (CCDF):

Show Chart: Selecting this button switches to a Chart view of the result. You can also toggle between the Distribution Summary View and the Chart View by double-clicking in the preview pane. Note that when viewing a Distribution Summary, both the Display Chart and the Display Table buttons appear deselected.

Read more: Viewing a Distribution Chart (page 647).

Show Table: Selecting this button switches to a Table view of the result. Note that when viewing a Distribution Summary, both the Display Chart and the Display Table buttons appear deselected.

Read more: Viewing a Distribution Table (page 651).

Show Confidence Bounds: If you press this button or press Ctrl+Shift+B, GoldSim will display confidence bounds on CDFs and CCDFs (they cannot be shown for PDFs):
These bounds reflect uncertainty in the probability distribution due to the finite number of Monte Carlo realizations (as the number of realizations is increased, the uncertainty in the distribution decreases). The confidence bounds represent the 5% and 95% confidence limits on the distribution. The calculation of confidence bounds is discussed in detail in Appendix B. Note that confidence bounds cannot be displayed if importance sampling has been used in the model.

**Show Result Array**: Pressing this button displays a new (modal) window containing the result array. The result array represents a sorted table of all of the results, and is used to produce the charts and statistics for the distribution. The table shows the value, the weight, the number of occurrences of the value, the cumulative probability of the value, and the 5% and 95% confidence bounds for the value at that cumulative probability. (Note, however, that confidence bounds cannot be computed if importance sampling has been used in the model.)

*Read more: Viewing the Distribution Result Array* (page 653).

**Edit Properties**: This provides access to the Result Properties.

*Read more: Viewing the Properties of a Distribution Result* (page 640).

**Note**: If you press the Copy button (or Ctrl+C) from the Summary View of a Distribution result, the preview pane chart is copied to the clipboard.

By default, Distribution results operate on Final Values. That is, the display uses the values at the end of each realization. However, by defining Capture Points, you can display results at any specified time. If you have created Capture Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the results at the specified Capture Point) that you would like to display.

*Read more: Viewing Results at Capture Points* (page 569).
Note: When viewing distribution results produced by a nested Monte Carlo simulation (using a SubModel), the Distribution Summary dialog is modified somewhat and provides slightly different options.

Read more: Carrying Out Nested Monte Carlo Simulation Using a SubModel (page 992).

When you display a Distribution Result by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element’s primary output) and selecting Distribution Result… from the context menu, by default, a Distribution Summary will be displayed.

Read more: Viewing a Distribution Summary (page 643).

You can view a Distribution Chart by pressing the Show Chart button when viewing a Distribution Summary or a Distribution Table:

You can also view a Distribution Chart by double-clicking on the preview pane in the Distribution Summary window.

Note: When viewing a Distribution Result element, the element “remembers” the last type of view that was displayed, and displays that view when you double-click on it.

A Distribution Chart looks like this:

The Distribution Chart has a variety of buttons at the top of the window:
The functions of these buttons are:

**Display PDF**: This displays the chart as a probability density function (PDF).

**Display CDF**: This displays the chart as a cumulative distribution function (CDF).

**Display CCDF**: This displays the chart as a complementary cumulative distribution function (CCDF).

**Show Chart**: Deselecting this button switches to a Distribution Summary view of the result. You can also toggle from the Chart View to the Distribution Summary View by double-clicking anywhere in the chart. Note that when viewing a Distribution Chart, the Display Chart button appears selected.

**Show Table**: Selecting this button switches to a Distribution Table view of the result. Note that when viewing a Distribution Table, the Display Table button appears selected.

**Read more**: Viewing a Distribution Table (page 651).

**Show Confidence Bounds**: If you press this button or press Ctrl+Shift+B, GoldSim will display confidence bounds on CDFs and CCDFs (they cannot be displayed for PDFs):
These bounds reflect uncertainty in the probability distribution due to the finite number of Monte Carlo realizations (as the number of realizations is increased, the uncertainty in the distribution decreases). The confidence bounds represent the 5% and 95% confidence limits on the distribution. The calculation of confidence bounds is discussed in detail in Appendix B. Note that confidence bounds cannot be displayed if importance sampling has been used in the model.

**Show Result Array:** Pressing this button displays a new (modal) window containing the result array. The result array represents a sorted table of all of the results, and is used to produce the charts and statistics for the distribution. The table shows the value, the weight, the number of occurrences of the value, the cumulative probability of the value, and the 5% and 95% confidence bounds for the value at that cumulative probability. (Note, however, that confidence bounds cannot be computed if importance sampling has been used in the model.)

**Read more:** Viewing the Distribution Result Array (page 653).

**Edit Chart Style:** This button provides access to a dialog for editing the chart style.

**Read more:** Controlling the Chart Style in Distribution Results (page 668).

**Edit Properties:** This provides access to the Result Properties.

**Read more:** Viewing the Properties of a Distribution Result (page 640).

Like all result charts, you can also control various attributes of the chart via a context menu. This includes the ability to turn on and off a legend. The legend uses the Label for the result defined in the Result Properties page. When viewing multiple results on one chart, you will always want to display the legend.

**Read more:** Using Context Menus in Charts (page 566); Viewing Distributions of Multiple Outputs (page 654).

When the results being plotted are conditions (i.e., True or False) or represent discrete (as opposed to continuous) distributions, GoldSim uses special algorithms to plot these in a consistent and effective manner.

**Read more:** Plotting Condition Distributions (page 650); Plotting Discrete Distributions (page 650).

You can copy the chart to the clipboard by right-clicking in the chart, and selecting Copy from the context menu. Alternatively, you can press Ctrl+C while viewing the chart. To export a chart, right-click in the chart, and select Export… from the context menu.

**Read more:** Copying a Chart or Table (page 734); Exporting a Chart (page 734).

By default, Distribution results operate on Final Values. That is, the display uses the values at the end of each realization. However, by defining Capture Points, you can display results at any specified time. If you have created Capture Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the results at the specified Capture Point) that you would like to display.

**Read more:** Viewing Results at Capture Points (page 569).
Note: When viewing distribution results produced by a nested Monte Carlo simulation (using a SubModel), the Distribution Chart display is modified somewhat and provides slightly different options.

Read more: Carrying Out Nested Monte Carlo Simulation Using a SubModel (page 992).

Condition outputs are either True or False and are typically used as state variables or flags in a simulation. As a result, in some situations, you may want to save and plot distributions of conditions. To facilitate this, when plotting a distribution of a condition, GoldSim plots True as 1 and False as 0.

When viewing a distribution of conditions in the Distribution Summary window, within the Percentile portion of the window, GoldSim does not display percentiles; rather it displays the probability of True and False:

The displayed Mean is actually computed as the probability of being True. The other statistics are not computed. GoldSim also displays the 5% and 95% confidence bounds on the Mean.

Read more: Understanding Output Attributes (page 87); Viewing a Distribution Summary (page 643).

Plotting Discrete Distributions

Discrete distributions only take on a finite number of discrete values (i.e., they are not continuous). GoldSim’s Monte Carlo engine includes algorithms that attempt to identify discrete (as opposed to continuous) results, and plot them as such.

When viewing a discrete distribution in the Distribution Summary window, within the Percentile portion of the window GoldSim does not display percentiles; rather, it displays the actual probability of each result:
When you display a distribution result by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element’s primary output) and selecting Distribution Result… from the context menu, by default, a Distribution Summary will be displayed.

**Read more:** Viewing a Distribution Summary (page 643).

You can view a Distribution Table by pressing the Show Table button when viewing a Distribution Summary or a Distribution Chart.

**Note:** When viewing a Distribution Result element, the element “remembers” the last type of view that was displayed, and displays that view when you double-click on it.

A Distribution Table looks like this:
Each row of the table is a separate realization. The header row for a table shows the units for the result above the first column and the Label for the result. The Label is specified in the Result Properties dialog and defaults to the result (output) name.

You can increase or decrease the width of a column by dragging the line separating columns to the right or to the left, respectively. If the column is too narrow to display the contents of a cell, holding the cursor over the cell displays a tool-tip with the contents.

The Distribution Table has a variety of buttons at the top of the window:

- **Display PDF**: Switches to Distribution Summary view and displays the preview pane as a probability density function (PDF).
- **Display CDF**: Switches to Distribution Summary view and displays the preview pane as a cumulative distribution function (CDF).
- **Display CCDF**: Switches to Distribution Summary view and displays the preview pane as a complementary cumulative distribution function (CCDF).
- **Show Chart**: Selecting this button switches to a Distribution Chart view of the result.
- **Show Table**: Deselecting this button switches to a Distribution Summary view of the result. Note that when viewing a Distribution Table, the Display Table button appears selected.
- **Sort**: This button allows you to sort the rows based on a selected column.
- **Show Result Array**: Pressing this button displays a new (modal) window containing the result array. The result array represents a sorted table of all of the results, and is used to produce the charts and statistics for the distribution. The table shows the value, the weight, the number of occurrences of the value, the cumulative probability of the value, and the 5% and 95% confidence bounds for the value at that cumulative probability. (Note, however, that confidence bounds cannot be computed if importance sampling has been used in the model.)
- **Edit Properties**: This provides access to the Result Properties.

Several additional points should be noted regarding Distribution Tables:

- Conditions are displayed in tables as either 1/0, true/false, True/False, TRUE/FALSE, on/off, or High/Low. You select which of these pairs to use on the Results tab of the Options dialog (accessed via Model | Options… from the main menu).
You can control the number of significant figures displayed in tables from the **Results** tab of the Options dialog (accessed via **Model | Options...** from the main menu).

**Read more:** [Controlling Significant Figures and Scientific Notation in Result Displays](page 568).

You can copy the contents of a distribution table to the clipboard by pressing **Ctrl+C**. You must first select the cells you wish to copy. You can do so by placing your cursor in one cell and dragging to another location. You can select the entire table by pressing the cell in the upper left-hand corner of the table. Cells that are not adjacent can be selected using **Ctrl** key when selecting. Selected items will be highlighted in black. You can subsequently paste the table into another application (such as a spreadsheet).

**Read more:** [Selecting Items and Copying Values in Result Tables](page 567).

By default, Distribution results operate on Final Values. That is, the display uses the values at the end of each realization. However, by defining Capture Points, you can display results at any specified time. If you have created Capture Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the results at the specified Capture Point) that you would like to display.

**Read more:** [Viewing Results at Capture Points](page 569).

GoldSim allows you to view an alternative kind of table view of a distribution result referred to as the **result array**. This table is accessed by pressing the **Show Result Array** button in any of the Distribution Result display windows:

A typical result array is shown below:

<table>
<thead>
<tr>
<th>#</th>
<th>Value</th>
<th>Weight</th>
<th>Count</th>
<th>Cum</th>
<th>Q05</th>
<th>Q95</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>170</td>
<td>0.01</td>
<td>1</td>
<td>0.005</td>
<td>112</td>
<td>141</td>
</tr>
<tr>
<td>2</td>
<td>133</td>
<td>0.01</td>
<td>1</td>
<td>0.015</td>
<td>112</td>
<td>157</td>
</tr>
<tr>
<td>3</td>
<td>148</td>
<td>0.01</td>
<td>1</td>
<td>0.025</td>
<td>112</td>
<td>157</td>
</tr>
<tr>
<td>4</td>
<td>158</td>
<td>0.01</td>
<td>1</td>
<td>0.035</td>
<td>122</td>
<td>173</td>
</tr>
<tr>
<td>5</td>
<td>160</td>
<td>0.01</td>
<td>1</td>
<td>0.045</td>
<td>131</td>
<td>181</td>
</tr>
<tr>
<td>6</td>
<td>168</td>
<td>0.055</td>
<td>1</td>
<td>0.055</td>
<td>140</td>
<td>191</td>
</tr>
<tr>
<td>7</td>
<td>172</td>
<td>0.01</td>
<td>1</td>
<td>0.065</td>
<td>149</td>
<td>195</td>
</tr>
<tr>
<td>8</td>
<td>176</td>
<td>0.01</td>
<td>1</td>
<td>0.075</td>
<td>156</td>
<td>202</td>
</tr>
<tr>
<td>9</td>
<td>184</td>
<td>0.01</td>
<td>1</td>
<td>0.085</td>
<td>158</td>
<td>208</td>
</tr>
<tr>
<td>10</td>
<td>191</td>
<td>0.01</td>
<td>1</td>
<td>0.095</td>
<td>163</td>
<td>212</td>
</tr>
<tr>
<td>11</td>
<td>194</td>
<td>0.01</td>
<td>1</td>
<td>0.105</td>
<td>169</td>
<td>217</td>
</tr>
<tr>
<td>12</td>
<td>198</td>
<td>0.01</td>
<td>1</td>
<td>0.115</td>
<td>172</td>
<td>222</td>
</tr>
<tr>
<td>13</td>
<td>205</td>
<td>0.01</td>
<td>1</td>
<td>0.125</td>
<td>175</td>
<td>227</td>
</tr>
<tr>
<td>14</td>
<td>209</td>
<td>0.01</td>
<td>1</td>
<td>0.135</td>
<td>180</td>
<td>232</td>
</tr>
<tr>
<td>15</td>
<td>212</td>
<td>0.01</td>
<td>1</td>
<td>0.145</td>
<td>188</td>
<td>236</td>
</tr>
<tr>
<td>16</td>
<td>216</td>
<td>0.01</td>
<td>1</td>
<td>0.155</td>
<td>192</td>
<td>238</td>
</tr>
<tr>
<td>17</td>
<td>220</td>
<td>0.01</td>
<td>1</td>
<td>0.165</td>
<td>194</td>
<td>244</td>
</tr>
<tr>
<td>18</td>
<td>224</td>
<td>0.01</td>
<td>1</td>
<td>0.175</td>
<td>196</td>
<td>247</td>
</tr>
</tbody>
</table>

The result array represents a sorted table of all of the results, and is used to produce the charts and statistics for the distribution. The table shows the value, the weight, the number of occurrences of the value, the cumulative probability of the value, and the 5% and 95% confidence bounds for the value at that cumulative probability. (Note, however, that confidence bounds cannot be computed if importance sampling has been used in the model.)
You can increase or decrease the width of a column by dragging the line separating columns to the right or to the left, respectively. If the column is too narrow to display the contents of a cell, holding the cursor over the cell displays a tool-tip with the contents.

Several additional points should be noted regarding Distribution Result Arrays:

- You can copy the contents of the table to the clipboard by pressing the cell in the upper left-hand corner (#) to select the entire table, and then pressing Ctrl+C to copy the contents to the clipboard. You can subsequently paste the table into another application (such as a spreadsheet).
- Values in the table, by definition, are sorted in ascending order. To sort in descending order, double-click on a column header twice. Double-click on a header again to return to ascending order.
- You can control the number of significant figures displayed in tables from the Results tab of the Options dialog (accessed via Model | Options… from the main menu).

Read more: Controlling Significant Figures and Scientific Notation in Result Displays (page 568).

The manner in which the results array is created and used is discussed in detail in Appendix B.

It is often useful to view the distributions of multiple outputs on a single chart or table. Displaying multiple distributions in this way allows you to more directly compare and contrast the results.

To display multiple results in a Distribution Result, you must first open the Result Properties dialog for the distribution.

Read more: Viewing the Properties of a Distribution Result (page 640).

You can do this by pressing the Edit Properties... button in any of the Distribution Result display windows:

The following dialog will be displayed:
Note that the Label is user-editable and is used in legends and column headers.

To add addition outputs to the result, press the Add Result… button. A dialog for selecting an output will be displayed. After you select the output you wish to add to the result display, and press OK, the selected outputs are then shown in the Result Properties dialog:

You can show the full path for each result (showing the containment hierarchy) by placing your cursor over the Result item to display a tool-tip.

The following points should be noted regarding the Distribution Result Properties dialog when displaying multiple results:

- The units of the results default to the Display Units of the first result added to the list. However, you can change the Display Units that are displayed from within the Result Properties dialog.
You can add as many results as desired to list. Note, however, that all results must have the same dimensions. GoldSim will not allow you to add an output which does not match the dimensions of the existing result(s).

Only those results in which the Show box is checked will be included in displays. Hence, after adding a result to the list, you can temporarily hide it from displays by clearing this box.

The Label is user-editable, and is used in legends and column headers.

The Style of each line displayed in a Distribution Chart can be edited by clicking on the field (in the Style column) corresponding to each result.

Read more: Controlling the Chart Style in Distribution Results (page 668).

Pressing the Show Result button displays the multiple output display.

The Distribution Summary looks like this:

Note that although the Preview Pane will show all results, the Distribution Summary can only show Percentiles and Statistics for one result at a time. You can choose which of the results to display from the drop list at the upper left-hand corner of the window.

Read more: Viewing a Distribution Summary (page 643).

A Distribution Chart looks like this:
Note that the legend displays the (user-editable) Labels defined for each result.

**Read more:** Viewing a Distribution Chart (page 647).

Note that a legend is available for Distribution Charts (you may have to turn it on by right-clicking in the chart and selecting to show the legend from the context menu). The Labels specified in the Result Properties dialog are used in the legend to label the different results.

For Distribution Tables, the results are listed in separate columns:

Note that the column headers are the (user-editable) Labels defined for each result.

**Read more:** Viewing a Distribution Table (page 651).

**Note:** You cannot show Confidence Bounds in the Distribution Chart, Table or the Summary when displaying multiple outputs (the Confidence Bounds button is grayed out).
A Distribution Result can also be used to view a single realization simulation. This is particularly useful if you want to compare different results (or scenarios) in a single display.

When displaying Distribution Results after running only a single realization, the Distribution Chart looks like this:

![Distribution Chart Example]

In this example, three results have been added to a Result element via the Properties dialog. Note that the legend displays the (user-editable) Labels defined for each result.

Read more: Viewing a Distribution Chart (page 647).

A legend is available for Distribution Charts (you may have to turn it on by right-clicking in the chart and selecting to show the legend from the context menu). The Labels specified in the Result Properties dialog are used in the legend to label the different results.

For Distribution Tables, the results are listed in separate rows:

![Distribution Table Example]

Note that the row headers are the (user-editable) Labels defined for each result.

Read more: Viewing a Distribution Table (page 651).

When carrying out probabilistic simulations, you may often run hundreds or thousands of realizations. In order to analyze the results, it is often quite useful to classify the realizations into categories. A category is simply defined by a condition relating one or more outputs in the model (e.g., those realizations in which the discount rate was above 3.5%; those realizations in which the profit exceeded $1,000,000; those realizations in which the peak concentration was between 1 mg/l and 10 mg/l).

Read more: Classifying and Screening Realizations (page 577).

Classification categories are defined at the bottom of the Monte Carlo Result Display Properties dialog:
This dialog can be accessed from the Monte Carlo tab of the Simulation Settings dialog (via the Result Options… button), and is also accessible by pressing the Options… button available in the Result Properties dialog of a Distribution Result element.

Once you have defined categories, Distribution results can be displayed by category. The Properties dialog for a Distribution Result with categories defined looks like this:

In addition to showing the result (Mass in this case), the three categories that have been defined are also shown.

**Note:** Categories are only shown if the Distribution Result contains single result. If you add multiple results, the categories will not be shown.

By default, the categories are **not** shown in result displays (the Show box is cleared). If you want to display the categories, you must check the box.

In this particular example, the net number of realizations falling into each category is as follows: 11.8% of the realizations fall into the Low category, 56% fall into the Medium category, and 32.2% fall into the High category. This information is displayed in the the Monte Carlo Result Display Properties dialog after you run the simulation.
When you display a Distribution Result with categories, however, the results are displayed in terms of the gross number (as indicated in the Result Properties dialog). The gross number represents the percentage of realizations falling into the selected category and all the categories above it. Hence, in this example, 11.8% of the realizations fall into the Low category, 67.8% fall into the Low or Medium category, and 100% fall into the Low, Medium or High category.

If you check the Show boxes for all categories, and press the Show Result button, the categories will be included in the displays.

Note that for each category, you can edit the Style via the Monte Carlo Result Display Properties dialog. (The Style selections for each category are displayed in the Distribution Result Properties dialog, but cannot be edited there.)

Read more: Controlling the Chart Style in Distribution Results (page 668).

The Distribution Summary with categories looks like this:

![Distribution Summary](image)

Note that although the Preview Pane will show all categories, the Distribution Summary can only show Percentiles and Statistics for one category at a time. You can choose which of the categories to display from the drop list at the upper left-hand corner of the window.

Read more: Viewing a Distribution Summary (page 643).

A Distribution Chart with categories looks like this:
The Labels specified in the Monte Carlo Result Display Properties dialog are used in the legend to label the category results. The label for the result itself (in this case Mass) is specified in the Result element dialog.

Note that a legend is available for Distribution Charts (you may have to turn it on by right-clicking in the chart and selecting to show the legend from the context menu).

Note that because a Distribution Result displays categories in terms of gross numbers, the final category (in this case High) actually includes all realizations, and hence is identical to the display of the result itself.

Read more: Viewing a Distribution Chart (page 647).

For Distribution Tables with categories, the results are listed in separate columns:
Note that only realizations that fall into each category are displayed. If a realization is not in a category, it is blank. Because a Distribution Result displays categories in terms of gross numbers (and categories are displayed in the table from left to right), if a realization falls into a category, it also falls into all category columns to the right.

Read more: Viewing a Distribution Table (page 651).

Note: You cannot show Confidence Bounds in the Distribution Chart, Table or the Summary when displaying categories (the Confidence Bounds button is grayed out).

Within all result displays, you can also choose to screen out one or more categories, so that the results that are shown (in charts and tables) only include those realizations in the categories which you have chosen to include. You can screen a particular category from the results by clearing the Include box at the bottom of the Monte Carlo Result Display Properties dialog:

Note: You can edit the Include box in Edit Mode, Result Mode Scenario Mode or Run Mode.

Result screening only affects the result itself (if you choose to show categories, the screening does not apply to the category displays). Hence, in the example below, we are screening Low and High categories from the result (labeled Mass), but simultaneously displaying the (gross) category results:
Screened results shown with category results (which are not screened). In most cases, you would likely not show category results when screening.

Note that when screening results, it is the net number of realizations that are screened. Hence, in the example above (in which the Low and High categories are screened), the screened result is based on only those results in the Medium category (56% of the realizations in this example).

In most cases, when you add an output to a Distribution Result (via the Add Result... button in the Result Properties dialog), you will simply be adding a standard output (such as the output of an Expression element or the primary output of a Reservoir element). However, several types of elements in GoldSim produce a specialized output referred to as a Distribution output, and this output type can also be added (and viewed) in a Distribution Result.

Distribution outputs are complex outputs that represent all the statistical information necessary to define a probability distribution. They can only be produced by Stochastic elements, SubModels and Spreadsheet elements.

Read more: Stochastic Elements (page 153); Creating the Output Interface to a SubModel (page 984); Importing Stochastic Element Definitions from a Spreadsheet (page 918).

These outputs can be clearly identified when viewing the output port of an element:
Note the Distribution output has a different appearance than the other three standard outputs (it is gray rather than blue).

In some situations, you may want to add such an output to a Distribution Result. There are two primary cases where this can be of value:

1. You wish to carry out a nested Monte Carlo simulation (using a SubModel). In order to view the results, you would add a Distribution output from a Monte Carlo SubModel to a Distribution Result element in the parent model.

   Read more: Carrying Out Nested Monte Carlo Simulation Using a SubModel (page 992).

2. You may want to compare a sampled distribution to a specified analytical distribution (e.g., to see how closely the sampled distribution matches a particular distribution such as a Normal or Weibull).

   This is possible because Stochastic elements always have a Distribution output and if you view the Distribution output for a Stochastic in a Distribution Result it will display the analytical distribution (as opposed to the sampled distribution for that Stochastic). In the example below, a Stochastic element is defined (as a Normal) and the model is run for 300 realizations. Two of its outputs are added to a Distribution Result: the primary output and the Distribution output:

   ![Distribution Result Properties](image)

   The Result display would look like this:
The primary output (Stochastic1) shows the sampled distribution. The Distribution output (Stochastic1.Distribution) shows the analytical distribution shape.

GoldSim’s scenario modeling capability allows you to directly compare results generated by different sets of input parameters. In effect, when you use this capability, your model can store (and subsequently compare) multiple sets of results (and inputs).

Read more: Creating, Running and Comparing Scenarios (page 504).

When a model is in Scenario Mode, Distribution Result elements can be used to view scenario results.

Note: Scenario results are only shown for the first result listed in the Distribution Result element. If the Distribution Result contains multiple results, in Scenario Mode only the first result will be displayed.

In order for scenario results to be displayed in a Distribution Result element, the Show button must be checked for each scenario to be displayed in the Scenario Manager (accessed from the main menu (Run | Scenario Manager…) or by pressing F7):

In this example, the Show box is checked for all three scenarios, so all three will be displayed in results.
If you double-click on a Distribution Result element in Scenario Mode, GoldSim displays results for all scenarios for which scenario results have been generated (and for which the **Show** button has been checked from within the Scenario Manager).

Note that for each scenario, within the Scenario Manager you can edit the **Style**, as well as the **Scenario Name**. These affect how the results are labeled in chart and table displays.

**Read more:** [Controlling the Chart Style in Distribution Results](#) (page 668).

If you have run multiple realizations for the various scenarios (and hence are in Scenario Mode), a Distribution Summary in Scenario Mode looks like this:

![Distribution Chart](image)

Note that although the Preview Pane will show all scenarios, the Distribution Summary can only show Percentiles and Statistics for one scenario at a time. You can choose which of the scenarios to display from the drop list at the upper left-hand corner of the window.

**Read more:** [Viewing a Distribution Summary](#) (page 643).

A Distribution Chart in Scenario Mode looks like this:
The Scenario Names specified in the Scenario Manager dialog are used in the legend (in parentheses) after the Result Label (specified in the Result element dialog).

**Read more:** Viewing a Distribution Chart (page 647).

For Distribution Tables in Scenario Mode, the scenarios are listed in separate columns:

**Read more:** Viewing a Distribution Table (page 651).

**Note:** You cannot show Confidence Bounds in the Distribution Chart, Table or the Summary in Scenario Mode (the Confidence Bounds button is grayed out).

When running and comparing scenarios, it is often useful to view Distribution Results even when running only a single realization. This is because when displaying Distribution Results after running only a single realization, GoldSim will display the single results for each scenario side-by-side in the form or a bar chart.

**Read more:** Viewing Distribution Results for Single Realization Runs (page 658).

For example, a Distribution Chart in Scenario Mode looks like this:
This particular chart shows three scenarios.

When viewing a Distribution Result in Scenario Mode, you cannot view results by category (i.e., based on the categories you may have defined at the bottom of the Monte Carlo Result Display Properties dialog):

That is, in Scenario Mode, results are never displayed by category. However, screening by category is active in Scenario Model. That is, if you have chosen to screen out one or more categories (by clearing the Include box in the dialog above), the scenario results that are shown (in charts and tables) only include those realizations in the categories which you have chosen to include.

Note: Depending on how you have defined the categories, it is possible for a realization to fall into a different category in different scenarios. Hence, when screening results in Scenario Mode, the number of realizations within each scenario may differ.

Read more: Using Result Classification and Screening in Distribution Results (page 658).

Controlling the Chart Style in Distribution Results

GoldSim has powerful charting capabilities that allow you to customize the appearance of each chart. This includes adding headers and footers, and changing axis scales and labels. Most of these attributes can be edited by pressing the Chart Style button at the top of the Distribution Chart window:
Pressing this button (or right-clicking in a chart and selecting **Edit Chart Style**) provides access to the following dialog for editing the various chart properties:

This dialog is common to all types of charts, and is discussed elsewhere.  

**Read more:** [Editing the Appearance of a Chart](page 702).

In addition to the basic chart attributes controlled by the Chart Style dialog, the Distribution Result Properties dialog itself is used to control some of the attributes of a chart:

- The units of the results default to the Display Units of the first result added to the list. However, you can change the **Display Units** that are displayed from within the Result Properties dialog.
• Only those results in which the Show box is checked will be included in displays. Hence, after adding a result to the list, you can temporarily hide it from displays by clearing this box.

• The Label is user-editable, and is used in legends (and column headers for tables).

Note: You can hide or show the legend on a chart via the context menu (i.e., accessed by right-clicking in the chart).

• The Style of each line displayed in a Distribution Chart can be edited by clicking on the field (in the Style column) corresponding to each result. This provides access to the following dialog for editing the line style (for a distribution) or the fill style (for a single realization display):

![Style Attributes dialog]

Read more: Editing Data Styles (page 710).

If you have defined realization categories, have run multiple realizations, and a Distribution Result element has only a single result defined, you will note that in addition to showing the result (Mass in the example below), any categories that have been defined are also shown:
Read more: Using Result Classification and Screening in Distribution Results (page 658).

By default, the categories are not shown in result displays (the Show box is cleared). If you want to display the categories, you must check the box.

Note that although the Style selections for each category are displayed in the Distribution Result Properties dialog, they cannot be edited there (if you try to do so, you will note that they are grayed out). Instead, the Style for each category is controlled via the Monte Carlo Result Display Properties dialog (most easily accessed by pressing the Options... button in the Result Properties dialog of a Distribution Result element). The bottom of the Monte Carlo Result Display Properties dialog looks like this:

The Style used for each category displayed in a Distribution Chart can be edited by clicking on the field (in the Style column) corresponding to each category.

When a model is in Scenario Mode, Distribution Result elements can be used to view scenario results. In this case, the line style for distribution results for the different scenarios is controlled by the Scenario Manager dialog (accessed from the main menu (Run | Scenario Manager...) or by pressing F7):
Viewing Multi-Variate Results

In order for scenario results to be displayed in a Distribution Result element, the Show button must be checked for each scenario to be displayed in the Scenario Manager.

Note that for each scenario, you can edit the Style, as well as the Scenario Name, which is used used in legends (and column headers for tables).

**Viewing Multi-Variate Results**

Multi-Variate results allow you to analyze and compare multiple outputs in graphical or tabular form. That is, by definition, a Multi-Variate result display requires at least two separate output variables (i.e., results). A Multi-Variate result display provides a mechanism for creating scatter plots, as well as carrying out sensitivity and correlation analysis among the selected outputs.

To view Multi-Variate results, you must select a specific output that you are interested in, along with one or more additional outputs that may have affected that result.

Five types of Multi-Variate result displays are available:

- 2D scatter plots;
- 3D scatter plots;
- Sensitivity analyses;
- Correlation analyses; and
- “Raw” Data display.

**Note:** Multi-Variate results are not available in Scenario Mode. They can only be viewed in Result Mode.

**Selecting Outputs for a Multi-Variate Result Display**

Unlike other result types, a Multi-Variate results requires specification of at least two outputs. If you have saved Final Values for an output, you can display a Multi-Variate result by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element's primary output) and selecting Multi-Variate Result... from the context menu.

**Read more:** Viewing Scenario Results in Distribution Result Elements (page 665).
The output that you selected will be defined as the result to evaluate. You will then be prompted to select an input variable (i.e., some other output in the model) upon which you wish to base your analysis (multi-variate result displays require at least one variable to be selected):

A browser will then appear allowing you to select a variable (i.e., another output in the model). After doing so, the default display for a Multi-Variate result (a 2D scatter plot) is displayed:

**Note:** By default, a legend will be turned on, labeling the different categories into which the realizations have been classified. If you have not defined categories, you will want to hide the legend (by right-clicking in the chart and clearing View | Show Legend).

**Read more:** Using Result Classification and Screening in Multi-Variate Results (page 686).

To add additional outputs to the Multi-Variate result, you must then press the Result Properties button. The Result Properties button is the furthest button to the right at the top of the display window:

When you do so, the Result Properties dialog for the Multi-Variate result will be displayed:
From this dialog, you can add additional variables (or delete variables) using the Add Result… and Delete Result buttons.

**Note:** If you delete variables such that you have less than two, you will not be able to display any results at all (since a Multi-Variate result requires at least two variables).

The Stochastics… button adds all Stochastic elements in the model to the list of variables. This is often of value, since when carrying out sensitivity or uncertainty analyses, the variables of interest are typically the Stochastics in your model.

For the purpose of creating scatter plots, the variables must be assigned to specific axes. The result (the output from which you selected Multi-Variate Result…) is assigned to the Y-axis (i.e., the “Result” axis). The first variable that you select is assigned to the X1-axis, and the second variable you select is assigned to the X2-axis. Additional variables are not assigned to any axis. This is indicated in the Result column of the dialog:
You can subsequently reassign these axes by moving variables up and down in the list using the **Move Up** and **Move Down** buttons.

If you click on the Result Properties button when viewing a Multi-Variate Result chart or table, the Properties dialog for the result will be displayed. The Result Properties button is the furthest button to the right at the top of the display window:

Note that when you press this button, it does not close the result display window you were displaying. Both windows are displayed simultaneously (although you can subsequently close one or the other). When viewing results, any changes you make to the Result Properties are immediately represented in the result display.

The Result Properties dialog is always modal. That is, it cannot be minimized, and with the exception of viewing the result display (which can be displayed simultaneously and share the focus with the Properties dialog), it retains the focus while it is displayed. As a result, you must close the Result Properties dialog before you can edit any other part of your GoldSim model.

The Result Properties dialog for an interactive Multi-Variate result looks like this:
At the top of the page is a button to **Create Element**. By pressing this button, a Result element is created. When you do so, the key change that will be noticeable is that the **Name** will now be editable. The **Create Element** button becomes an **Appearance** button (which is used to modify the appearance of the element itself).

**Read more**: [Creating and Using Result Elements](#) (page 570).

The **Show Result >>** button opens a result display window (while keeping the Properties dialog simultaneously open).

**Read more**: [Viewing a 2D Scatter Plot](#) (page 677); [Viewing a 3D Scatter Plot](#) (page 679); [Viewing a Sensitivity Analysis Table](#) (page 681); [Viewing a Correlation Matrix Table](#) (page 683); [Viewing a Raw Multi-Variate Data Table](#) (page 684).

The **Add Result...** button allows you to add other results to the chart. Results can be deleted with the **Delete Result** button. The order in which multiple results appear in the list can be changed using the **Move Up** and **Move Down** buttons.

The **Stochastics...** button adds all Stochastic elements in the model to the list of variables. This is often of value, since when carrying out sensitivity or uncertainty analyses, the variable of interest are typically the Stochastics in your model.

For the purpose of creating scatter plots, the variables must be assigned to specific axes. The first result in the list is assigned to the Y-axis (i.e., the “Result” axis). The second result is assigned to the X1-axis, and the third result is assigned to the X2-axis. Additional variables are not assigned to any axis. This is indicated in the Result column of the dialog.

**Read more**: [Selecting Outputs for a Multi-Variate Result Display](#) (page 672).

The Properties dialog for a Multi-Variate Result element has several differences:
The Create Element button is replaced by an Appearance button (which is used to modify the appearance of the element itself).

The Name (and Description) are editable (since this is an element).

An Options… button is available that provides access to the Monte Carlo Result Display Properties.

Read more: Controlling Monte Carlo Result Options (page 482).

A 2D Scatter Plot can provide a powerful visual image of the relationship between two variables.

If you have saved Final Values for an output, you can display a 2D Scatter Plot by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element's primary output) and selecting Multi-Variate Result… from the context menu. The output that you selected will be defined as the result to evaluate. You will then be prompted to select an input variable (i.e., some other output in the model) upon which you wish to base your analysis (multi-variate result displays require at least one variable to be selected).

A browser will then appear allowing you to select a variable (i.e., another output in the model). After doing so, the default display for a Multi-Variate result (a 2D scatter plot) is displayed:
If you are viewing a different type of display (either a chart or table), you can view a 2D Scatter Plot by pressing the Chart or 2D Chart button at the top of the display.

**Note:** By default, a legend will be turned on, labeling the different categories into which the realizations have been classified. If you have not defined categories, you will want to hide the legend (by right-clicking in the chart and clearing **View | Show Legend**).

**Note:** If only two variables are listed in the Result Properties dialog, only a 2D Chart can be shown, and hence only one button will be available (Chart). If three or more variables are present in the Result Properties dialog, buttons for both a 2D Chart and a 3D Chart become available.

**Note:** When viewing a Multi-Variate Result element, the element “remembers” the last type of view that was displayed, and displays that view when you double-click on it.

For the purpose of creating scatter plots, the variables must be assigned to specific axes. The first result in the list of results in the Result Properties dialog is assigned to the Y-axis (i.e., the “Result” axis). The second result is assigned to the X1-axis, and the third result is assigned to the X2-axis. Additional variables are not assigned to any axis. This is indicated in the Result column of the Result Properties dialog. The order in which multiple results appear in the list can be changed using the Move Up and Move Down buttons in that dialog.

**Read more:** [Selecting Outputs for a Multi-Variate Result Display](page 672).

The Use Ranks button determines whether the display uses ranks of the values or the actual values. By default, this button is not pressed, and the display uses actual values.
You can copy the chart to the clipboard by right-clicking in the chart, and selecting Copy from the context menu. Alternatively, you can press Ctrl+C while viewing the chart. To export a chart, right-click in the chart, and select Export… from the context menu.

Read more: Copying a Chart or Table (page 734); Exporting a Chart (page 734).

By default, Multi-Variate results operate on Final Values. That is, the display uses the values at the end of each realization. However, by defining Capture Points, you can display values at any specified time. If you have created Capture Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the values at the specified Capture Point) that you would like to display.

Read more: Viewing Results at Capture Points (page 569).

If you are using Result Classification, the plots points will be displayed in a different color for each category that has been defined.

Read more: Using Result Classification and Screening in Multi-Variate Results (page 686).

A 3D Scatter Plot (or a Cloud Plot) allows you to visualize the relationship between three variables.

The default view for a Multi-Variate result is a 2D Scatter Plot.

Read more: Viewing a 2D Scatter Plot (page 677).

If you are viewing a 2D Scatter Plot (or a different type of Multi-variate display), you can view a 3D Scatter Plot by pressing the 3D Chart button at the top of the display.

Note: Depending on how many variables you have defined in the Result Properties dialog, the 3D Chart button may not be available. In particular, in order to display a 3D Scatter Plot, you must have at least three results defined.

Read more: Viewing the Properties of a Multi-Variate Result (page 675).

Note: When viewing a Multi-Variate Result element, the element “remembers” the last type of view that was displayed, and displays that view when you double-click on it.

A 3D Scatter Plot looks like this:
For the purpose of creating scatter plots, the variables must be assigned to specific axes. The first output in the list of results in the Result Properties dialog is assigned to the Y-axis (i.e., the “Result” axis). The second output is assigned to the X1-axis, and the third output is assigned to the X2-axis. Additional variables are not assigned to any axis. This is indicated in the Result column of the Result Properties dialog. The order in which multiple results appear in the list can be changed using the Move Up and Move Down buttons in that dialog.

Read more: Selecting Outputs for a Multi-Variate Result Display (page 672).

The Use Ranks button determines whether the display uses ranks of the values or the actual values. By default, this button is not pressed, and the display uses actual values.

You can rotate the three dimensional image by holding down the left mouse button and dragging within the chart. You can also use the Up, Down Left and Right cursor keys.

In addition, 3D Scatter Plots have several hot-keys which can be used to modify the plot:

<table>
<thead>
<tr>
<th>Keys</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl+D</td>
<td>Toggles the drop lines (the lines drawn from each point to the base of the axis) on and off.</td>
</tr>
<tr>
<td>Ctrl+S</td>
<td>Toggles between a 3D scatter plot and a 2D scatter plot.</td>
</tr>
</tbody>
</table>

These options are also available from the context menu for the chart.

You can copy the chart to the clipboard by right-clicking in the chart, and selecting Copy from the context menu. Alternatively, you can press Ctrl+C while viewing the chart. To export a chart, right-click in the chart, and select Export... from the context menu.

Read more: Copying a Chart or Table (page 734); Exporting a Chart (page 734).

By default, Multi-Variate results operate on Final Values. That is, the display uses the values at the end of each realization. However, by defining Capture Points, you can display values at any specified time. If you have created Capture Points:
Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the values at the specified Capture Point) that you would like to display.

**Read more:** Viewing Results at Capture Points (page 569).

If you are using Result Classification, the plots points will be displayed in a different color for each category that has been defined.

**Read more:** Using Result Classification and Screening in Multi-Variate Results (page 686).

One of the most powerful options for analyzing multi-variate results is to carry out a sensitivity analysis. GoldSim provides a number of statistical sensitivity analyses through the multi-variate result display option.

The default view for a Multi-Variate result is a 2D Scatter Plot.

**Read more:** Viewing a 2D Scatter Plot (page 677).

If you are viewing a 2D Scatter Plot (or a different type of Multi-variate display), you can view a Sensitivity Analysis Table by pressing the **Sensitivity** button at the top of the display.

**Note:** When viewing a Multi-Variate Result element, the element “remembers” the last type of view that was displayed, and displays that view when you double-click on it.

A Sensitivity Analysis Table looks like this:

![Sensitivity Analysis Table](image)

This table displays measures of the sensitivity of the first output in the list of results in the Result Properties dialog (listed at the top of the dialog) to the selected input variables (all the other outputs in the list of results in the Result Properties dialog). In the example above, the analysis is being carried out on the result R1, and the other results whose impact on R1 is being measured are A, B, C and W.

**Note:** If some of your Stochastics are triggered repeatedly during a simulation, selecting them for sensitivity analysis is likely not to produce meaningful results.

The order in which results appear in the list in the Result Properties dialog can be changed using the **Move Up** and **Move Down** buttons in that dialog, such that any variable in the list can be specified as the result for which the sensitivity analysis is being carried out.

**Read more:** Selecting Outputs for a Multi-Variate Result Display (page 672).
Each of the measures computed in the Sensitivity Analysis Table is described below:

**Coefficient of determination:** This coefficient varies between 0 and 1, and represents the fraction of the total variance in the result that can be explained based on a linear (regression) relationship to the input variables (i.e., \( \text{Result} = aX + bY + cZ + \ldots \)). The closer this value is to 1, the better the relationship between the result and the variables can be explained with a linear model.

**Importance Measure:** This measure varies between 0 and 1, and represents the fraction of the result’s variance that is explained by the variable. This measure is useful in identifying nonlinear, non-monotonic relationships between an input variable and the result (which conventional correlation coefficients may not reveal). The importance measure is a normalized version of a measure discussed in Saltelli and Tarantola (2002).

**Correlation Coefficient:** Rank (Spearman) or value (Pearson) correlation coefficients range between -1 and 1, and express the extent to which there is a linear relationship between the selected result and an input variable.

**SRC (Standardized Regression Coefficient):** Standardized regression coefficients range between -1 and 1 and provide a normalized measure of the linear relationship between variables and the result. They are the regression coefficients found when all of the variables (and the result) are transformed and expressed in terms of the number of standard deviations away from their mean. GoldSim’s formulation is based on Iman et al (1985).

**Partial Correlation Coefficient:** Partial correlation coefficients vary between -1 and 1, and reflect the extent to which there is a linear relationship between the selected result and an input variable, after removing the effects of any linear relationships between the other input variables and both the result and the input variable in question. For systems where some of the input variables may be correlated, the partial correlation coefficients represent the “unique” contribution of each input to the result. GoldSim’s formulation is based on Iman et al (1985).

A more detailed discussion of how these measures are computed (along with full references) is presented in Appendix B.

You can copy the contents of a sensitivity analysis table to the clipboard. To do so, you must first select the entire table (by double-clicking on the empty cell in the upper left-hand corner of the table). After you do so, you can copy the table to the clipboard by Ctrl+C. You can subsequently paste the table into another application (such as a spreadsheet).

Like all result tables in GoldSim, this table can be sorted in ascending or descending order by selecting a column and pressing the **Sort** button:

![Sort button](image)

**Read more:** Sorting Values in Result Tables (page 568).

The **Use Ranks** button determines whether the calculations use ranks of the values or the actual values. By default, this button is not pressed, and the calculations are based on actual values.
By default, Multi-Variate results operate on Final Values. That is, the analysis applies to the values at the end of each realization. However, by defining Capture Points, you can carry out the analysis at any specified time. If you have created Capture Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the values at the specified Capture Point) that you would like to analyze.

Read more: Viewing Results at Capture Points (page 569).

Note: The sensitivity analyses presented here are statistical measures computed by analyzing multiple realizations of the model in which all of the Stochastic variables are simultaneously sampled each realization. GoldSim also provides a second type of sensitivity analysis in which you can vary one variable at a time, while holding all other variables constant.

Read more: Running Sensitivity Analyses (page 539).

Note: If you need to carry out more advanced sensitivity analyses, you can do so by exporting all results and using a third party analysis tool.

Read more: Exporting Final Value Results for Multiple Outputs and Multiple Realizations (page 733).

Viewing a Correlation Matrix Table

A Correlation Matrix tabulates the statistical correlation between two or more variables.

The default view for a Multi-Variate result is a 2D Scatter Plot.

Read more: Viewing a 2D Scatter Plot (page 677).

If you are viewing a 2D Scatter Plot (or a different type of Multi-variate display), you can view a Correlation Matrix Table by pressing the Correlations button at the top of the display.

Note: When viewing a Multi-Variate Result element, the element “remembers” the last type of view that was displayed, and displays that view when you double-click on it.

A Correlation Matrix Table looks like this:

![Correlation Matrix Table]

The correlation matrix consists of a table whose column and row headers are identical, and consist of all of the selected variables. The table entry for a particular row/column pair is the correlation coefficient for that pair of variables. The correlation coefficient takes on a value between –1 and 1, 1 being a perfect positive correlation, -1 being a perfect negative correlation.
This is the same analysis presented in the Sensitivity Analysis Table (in the Correlation Coefficient column), but is presented in terms of a matrix here, showing how all variables are correlated to each of the others. The Sensitivity Analysis table simply shows how a single result variable is correlated to the others (i.e., it shows one column of the matrix).

Read more: Viewing a Sensitivity Analysis Table (page 681).

A detailed discussion of how correlation coefficients are computed is presented in Appendix B.

You can increase or decrease the width of a column by dragging the line separating columns to the right or to the left, respectively. If the column is too narrow to display the contents of a cell, holding the cursor over the cell displays a tool-tip with the contents.

You can copy the contents of a correlation matrix table to the clipboard. To do so, you must first select the entire table (by double-clicking on the empty cell in the upper left-hand corner of the table). After you do so, you can copy the table to the clipboard by Ctrl+C. You can subsequently paste the table into another application (such as a spreadsheet).

Like all result tables in GoldSim, this table can be sorted in ascending or descending order by selecting a column and pressing the Sort button.

Read more: Sorting Values in Result Tables (page 568).

The Use Ranks button determines whether the calculations use ranks of the values or the actual values. By default, this button is not pressed, and the calculations are based on actual values.

By default, Multi-Variate results operate on Final Values. That is, the analysis applies to the values at the end of each realization. However, by defining Capture Points, you can carry out the analysis at any specified time. If you have created Capture Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the values at the specified Capture Point) that you would like to analyze.

Read more: Viewing Results at Capture Points (page 569).

In some situations, it is valuable to simply view a “raw” table of all the data values (i.e., for each realization) for each output specified in a Multi-Variate result.

The default view for a Multi-Variate result is a 2D Scatter Plot.

Read more: Viewing a 2D Scatter Plot (page 677).

If you are viewing a 2D Scatter Plot (or a different type of Multi-variate display), you can view a Multi-Variate Data Table by pressing the Table button at the top of the display.

Note: When viewing a Multi-Variate Result element, the element “remembers” the last type of view that was displayed, and displays that view when you double-click on it.

A Multi-Variate Data Table looks like this:
Each row of the table is a different realization, while the columns show the values of the selected outputs.

You can increase or decrease the width of a column by dragging the line separating columns to the right or to the left, respectively. If the column is too narrow to display the contents of a cell, holding the cursor over the cell displays a tool-tip with the contents.

Like all result tables in GoldSim, this table can be sorted in ascending or descending order by selecting a column and pressing the **Sort** button.

**Read more:** [Sorting Values in Result Tables](page 568).

You can copy the contents of the table to the clipboard. To do so, you must first select the entire table (by double-clicking on the empty cell in the upper left-hand corner of the table). After you do so, you can copy the table to the clipboard by **Ctrl+C**. You can subsequently paste the table into another application (such as a spreadsheet).

**Note:** You can control the number of significant figures displayed in tables from the Results tab of the Options dialog (accessed via **Model Options…** from the main menu).

**Read more:** [Controlling Significant Figures and Scientific Notation in Result Displays](page 568).

By default, Multi-Variate results operate on Final Values. That is, the display applies to the values at the end of each realization. However, by defining Capture Points, you can display values at any specified time. If you have created Capture Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the values at the specified Capture Point) that you would like to display.

**Read more:** [Viewing Results at Capture Points](page 569).

In some cases, you may want to export all of the data in a Multi-Variate Data Table to a file so that it can be read into another program (e.g., a statistical post-processor). GoldSim provides a mechanism to do this directly from the table by pressing a key combination.

**Read more:** [Exporting Final Value Results for Multiple Outputs and Multiple Realizations](page 733).
When carrying out probabilistic simulations, you may often run hundreds or thousands of realizations. In order to analyze the results, it is often quite useful to classify the realizations into categories. A category is simply defined by a condition relating one or more outputs in the model (e.g., those realizations in which the discount rate was above 3.5%; those realizations in which the profit exceeded $1,000,000; those realizations in which the peak concentration was between 1 mg/l and 10 mg/l).

Read more: Classifying and Screening Realizations (page 577).

Classification categories are defined at the bottom of the Monte Carlo Result Display Properties dialog:

This dialog can be accessed from the Monte Carlo tab of the Simulation Settings dialog (via the Result Options… button), and is also accessible by pressing the Options… button available in the Result Properties dialog of a Multi-Variate Result element.

When viewing a Multi-Variate result, you can display a 2-D or 3-D scatter plot by selecting 2D Plot or 3D Plot button at the top of the display.

Read more: Viewing a 2D Scatter Plot (page 677); Viewing a 3D Scatter Plot (page 679).

In a scatter plot, each point represents a different realization. When viewing a scatter plot in a model in which categories have been defined, each category is displayed in a different color (and/or symbol).

For the example categories shown above, the plot would look like this:
As can be seen, each category is presented as a different data set. Note that a legend is available for scatter plots (you may have to turn it on by right-clicking in the chart and selecting to show the legend from the context menu). The Labels specified in the Monte Carlo Result Display Properties dialog are used in the legend to label the category results.

The plot is based on the net number of realization falling into each category. In this particular example, the net number of realizations falling into each category is as follows: 63% of the realizations fall into the Low category, 24% fall into the Medium category, and 13% fall into the High category. This information is displayed in the Monte Carlo Result Display Properties dialog after you run the simulation.

Note that if you place your cursor over any of the points, it will show a tool-tip with information regarding that particular data point:

Note that for each category, you can edit the Style via the Monte Carlo Result Display Properties dialog.

Read more: Controlling the Chart Style in Multi-Variate Results (page 688).

Within all Multi-Variate result displays, you can also choose to screen out one or more categories, so that the results that are shown (in charts and tables) only include those realizations in the categories which you have chosen to include. You can screen a particular category from the results by clearing the Include box at the bottom of the Monte Carlo Result Display Properties dialog:
Viewing Multi-Variate Results

Realization Classification and Screening

Each Monte Carlo realization is assigned to the first category for which its final values satisfy the condition. Specify a condition for each category and select which categories to include in result analyses and displays.

<table>
<thead>
<tr>
<th>Include</th>
<th>Label</th>
<th>Condition</th>
<th>Style</th>
<th>Gross %</th>
<th>Net %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Low</td>
<td>B &lt; 10</td>
<td></td>
<td>63</td>
<td>63</td>
</tr>
<tr>
<td></td>
<td>Medium</td>
<td>B &lt; 20</td>
<td></td>
<td>87</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>All realizations</td>
<td></td>
<td>100</td>
<td>13</td>
</tr>
</tbody>
</table>

Note: You can edit the Include box in Edit Mode, Result Mode Scenario Mode or Run Mode.

In the example above, the Low and High categories have been screened out, so when viewing a 2D Scatter Plot, only the Medium category is displayed:

GoldSim has powerful charting capabilities that allow you to customize the appearance of each chart. This includes adding headers and footers, and changing axis scales and labels for scatter plots. Most of these attributes can be edited by pressing the Edit Chart Style button at the top of the Multi-Variate display window:

Pressing this button (or right-clicking in a scatter plot and selecting Edit Chart Style...) provides access to the following dialog for editing the various chart properties:

Controlling the Chart Style in Multi-Variate Results

Note that if you screened categories in this way and viewed a Sensitivity Analysis, Correlation Matrix or Data Table, the analysis or display is based only on the unscreened realizations.
Viewing Array Results

This dialog is common to all types of charts, and is discussed elsewhere.

Read more: Editing the Appearance of a Chart (page 702).

In the absence of Classification categories, the symbol used for a scatter plot is fixed and cannot be edited (a solid red square).

If you have defined realization categories, have run multiple realizations, then the symbol styles for the various categories can be specified in the Monte Carlo Result Display Properties dialog (most easily accessed by pressing the Options… button in the Result Properties dialog of a Distribution Result element). The bottom of the Monte Carlo Result Display Properties dialog looks like this:

Read more: Using Result Classification and Screening in Distribution Results (page 658).

The Style used for each category displayed in a scatter plot can be edited by clicking on the field (in the Style column) corresponding to each category.

Viewing Array Results

Many complex models utilize arrays (vectors and matrices).

Read more: Using Vectors and Matrices (page 782).

Array results allow you to view “snapshots” of vectors and matrices at particular points in time in both graphical and tabular form. By default, Array
results display Final Values (i.e., values at the end of each realization). However, by defining Capture Points, you can display results at any specified time. If you have created Capture Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the values at the specified Capture Point) that you would like to display.

*Read more:* Viewing Results at Capture Points (page 569).

If you have saved Final Values for an array output, you can display an Array result by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element's primary output) and selecting **Array Result…** from the context menu. By default, an Array Chart will be displayed:

![Array Chart Example]

*This example shows a vector chart. The vector has four items.*

---

**Note:** When viewing an Array Result element, the element “remembers” the last type of view that was displayed, and displays that view when you double-click on it.

**Note:** Array results are not available in Scenario Mode. They can only be viewed in Result Mode.

---

**Viewing the Properties of an Array Result**

If you click on the Result Properties button at the top of the display window when viewing an Array Result chart or table, the Properties dialog for the result will be displayed:

![Result Properties Button]

Note that when you press this button, it does not close the result display window you were displaying. Both windows are displayed simultaneously (although you can subsequently close one or the other). When viewing results, any changes you make to the Result Properties are immediately represented in the result display.
The Result Properties dialog is always modal. That is, it cannot be minimized, and with the exception of viewing the result display (which can be displayed simultaneously and share the focus with the Properties dialog), it retains the focus while it is displayed. As a result, you must close the Result Properties dialog before you can edit any other part of your GoldSim model.

The Result Properties dialog for an interactive Array result looks like this:

At the top of the page is a button to Create Element. By pressing this button, a Result element is created. When you do so, the key change that will be noticeable is that the Name will now be editable. The Create Element button becomes an Appearance button (which is used to modify the appearance of the element itself).

Read more: Creating and Using Result Elements (page 570).

The Show Result >> button opens a result display window (while keeping the Properties dialog simultaneously open).

Read more: Viewing a Vector Chart (page 692); Viewing a Matrix Chart (page 694); Viewing an Array Table (page 697).

The output being displayed by the Array result is listed directly to the right of the Array Result text. If you click on the output, it will open a browser, allowing you to select a different array (you must select an array; GoldSim will not allow you to select a scalar output). If you click on the Remove button (the red script X just above the Columns… button), it will remove the selected output (and replace it with “<Click to select>”). You would then need to click this to select another array.

The Label defaults to the name of the result being displayed. However, you can edit this (to make it more readable). It is displayed on the vertical axis of charts.

The Rows… and Columns… buttons provide access to a dialog for selecting which items to show in the display (and in the case of matrices, the Columns… button also provides access to a dialog for selecting the style of the bars shown in the chart). The Columns… button is only available for matrices.

Read more: Controlling the Chart Style in Array Results (page 700).

The Properties dialog for an Array Result element has several differences:
The Create Element button is replaced by an Appearance button (which is used to modify the appearance of the element itself).

- The Name (and Description) are editable (since this is an element).
- An Options… button is available that provides access to the Monte Carlo Result Display Properties.

Read more: Controlling Monte Carlo Result Options (page 482).

If you have saved Final Values for a vector output, you can display the final values of the vector by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element's primary output) and selecting Array Result… from the context menu.

By default, a Vector Chart will be displayed:

If you are viewing an Array Table, you can view a Vector Chart by pressing the Chart button at the top of the display.

**Note:** When viewing an ArrayResult element, the element “remembers” the last type of view that was displayed, and displays that view when you double-click on it.
By pressing the **Rows...** button from the Result Properties dialog for the Array result, you can select which items in the array are to be displayed (by default, all are displayed).

**Read more:** Viewing the Properties of an Array Result (page 690); Controlling the Chart Style in Array Results (page 700).

A row/column totals button is available at the top of the display window:

![Calculate and display row/column totals](image)

Pressing this button displays an additional bar with the total of all the row values:

![Fruit Production chart](image)

Vector charts have several hot-keys which can be used to modify the plot:

<table>
<thead>
<tr>
<th>Keys</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl+S</td>
<td>Switches between a bar chart and line chart.</td>
</tr>
<tr>
<td>Ctrl+W</td>
<td>Increases the width of the bars in a bar chart.</td>
</tr>
<tr>
<td>Ctrl-Shift+W</td>
<td>Decreases the width of the bars in a bar chart.</td>
</tr>
</tbody>
</table>

The following six hot-keys allow you to rotate and view the chart in three dimensions:

<table>
<thead>
<tr>
<th>Key</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>F6</td>
<td>Increases the depth (into the page) of the bars.</td>
</tr>
<tr>
<td>Shift+F6</td>
<td>Decreases the depth (into the page) of the bars.</td>
</tr>
<tr>
<td>F7</td>
<td>Increases the vertical inclination from which the chart is viewed.</td>
</tr>
<tr>
<td>Shift+F7</td>
<td>Decreases the vertical inclination from which the chart is viewed.</td>
</tr>
<tr>
<td>F8</td>
<td>Rotates the chart to the left.</td>
</tr>
<tr>
<td>Shift+F8</td>
<td>Rotates the chart to the right.</td>
</tr>
</tbody>
</table>

You can copy the chart to the clipboard by right-clicking in the chart, and selecting **Copy** from the context menu. Alternatively, you can press **Ctrl+C**.
while viewing the chart. To export a chart, right-click in the chart, and select Export… from the context menu.

**Read more:** Copying a Chart or Table (page 734); Exporting a Chart (page 734).

By default, Array results operate on Final Values. That is, the display uses the values at the end of each realization. However, by defining Capture Points, you can display values at any specified time. If you have created Capture Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the values at the specified Capture Point) that you would like to display.

**Read more:** Viewing Results at Capture Points (page 569).

If you have saved Final Values for a matrix output, you can display the final values of the matrix by right-clicking on the output (in a browser or the output interface), or the element (if the output is the element’s primary output) and selecting Array Result… from the context menu.

By default, a Matrix Chart will be displayed:

![Matrix Chart Example](image)

The row items are plotted along the X axis, with multiple bars for each column item (which are identified in a legend, which you may have to turn on by right-clicking in the chart and selecting to show the legend from the context menu).

If you are viewing an Array Table, you can view an Array Chart by pressing the 2D Chart button at the top of the display.

**Note:** When viewing an ArrayResult element, the element “remembers” the last type of view that was displayed, and displays that view when you double-click on it.

Matrix charts have several hot-keys which can be used to modify the plot:

<table>
<thead>
<tr>
<th>Keys</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl+S</td>
<td>Toggles between a bar chart and line chart.</td>
</tr>
<tr>
<td>Keys</td>
<td>Action</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>Ctrl+W</td>
<td>increases the width of the bars in a bar chart.</td>
</tr>
<tr>
<td>Ctrl-Shift+W</td>
<td>decreases the width of the bars in a bar chart.</td>
</tr>
</tbody>
</table>

The following six hot-keys allow you to rotate and view the chart in three dimensions:

<table>
<thead>
<tr>
<th>Keys</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>F6</td>
<td>increases the depth of the bars.</td>
</tr>
<tr>
<td>Shift+F6</td>
<td>decreases the depth of the bars.</td>
</tr>
<tr>
<td>F7</td>
<td>increases the vertical inclination from which the chart is viewed.</td>
</tr>
<tr>
<td>Shift+F7</td>
<td>decreases the vertical inclination from which the chart is viewed.</td>
</tr>
<tr>
<td>F8</td>
<td>rotates the chart to the left.</td>
</tr>
<tr>
<td>Shift+F8</td>
<td>rotates the chart to the right.</td>
</tr>
</tbody>
</table>

Pressing the **3D Chart** button displays the chart in three dimensions:

![3D Chart Example](image)

**Note**: In order for the labels to be legible for a 3D Chart, you will likely have to expand the window. The axes labels scale with the size of the window in 3D charts.

You can rotate the three dimensional image using by holding down the left mouse button and dragging within the chart. The following hot-keys can be used to manipulate the 3D Chart:

<table>
<thead>
<tr>
<th>Keys</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>F6</td>
<td>increases the depth (into the page) of the bars.</td>
</tr>
</tbody>
</table>
By pressing the **Rows...** and/or **Columns...** buttons from the Result Properties dialog for the array result, you can select which items in the array are to be displayed (by default, all are displayed). You can also change the colors for the column bars.

*Read more:* [Viewing the Properties of an Array Result](page 690); [Controlling the Chart Style in Array Results](page 700).

A row/column totals button is available at the top of the display window:

![Calculate and display row/column totals](image)

Pressing this button displays the row and column totals:

![Fruit Harvest](image)

You can copy the chart to the clipboard by right-clicking in the chart, and selecting **Copy** from the context menu. Alternatively, you can press **Ctrl+C** while viewing the chart. To export a chart, right-click in the chart, and select **Export...** from the context menu.

*Read more:* [Copying a Chart or Table](page 734); [Exporting a Chart](page 734).

By default, Array results operate on Final Values. That is, the display uses the values at the end of each realization. However, by defining Capture Points, you can display values at any specified time. If you have created Capture Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the values at the specified Capture Point) that you would like to display.

*Read more:* [Viewing Results at Capture Points](page 569).
Condition outputs are either True or False and are typically used as state variables or flags in a simulation. You can create arrays of conditions, and, in some situations, you may want to plot these arrays. To facilitate this, when plotting an array of a condition, GoldSim plots True as 1 and False as 0.

**[Read more: Understanding Output Attributes](#)** (page 87).

The default view for an Array result is a 2D Chart.

**[Read more: Viewing a Vector Chart](#)** (page 692); **[Viewing a Matrix Chart](#)** (page 694).

In some cases, however, you may want to view a table of values. If you are viewing a 2D Chart (or 3D Chart), you can view an Array Table by pressing the **Table** button at the top of the display.

---

**Note:** When viewing an ArrayResult element, the element “remembers” the last type of view that was displayed, and displays that view when you double-click on it.

---

An Array Table for a vector looks like this:

![Array Table for a vector](image)

The first column shows the item names for the vector. The units for the vector are in the first cell in the upper left-hand corner of the table.

An Array Table for a matrix looks like this:

![Array Table for a matrix](image)

The units for the vector are in the first cell in the upper left-hand corner of the table. The first row shows the item names for the columns. The first column shows the item names for the rows.

You can increase or decrease the width of a column by dragging the line separating columns to the right or to the left, respectively.

**Note:** Conditions are displayed in tables as either 1/0, true/false, True/False, TRUE/FALSE, on/off, or High/Low. You select which of these pairs to use on the Results tab of the Options dialog (accessed via **Model [Options...** from the main menu). The default setting is true/false.
Note: You can control the number of significant figures displayed in tables from the Results tab of the Options dialog (accessed via Model | Options… from the main menu).

Read more: Controlling Significant Figures and Scientific Notation in Result Displays (page 568).

By pressing the Rows… and/or Columns… buttons from the Result Properties dialog for the Array result, you can select which items in the array are to be displayed in the table (by default, all are displayed).

Read more: Viewing the Properties of an Array Result (page 690).

A row/column totals button is available at the top of the display window:

Pressing this button displays the column and row (for matrices) totals in the table:

![Array Result Display](image)

You can copy the contents of an array table to the clipboard by pressing Ctrl+C. You must first select the cells you wish to copy. You can do so by placing your cursor in one cell and dragging to another location. You can select the entire table by pressing the cell in the upper left-hand corner of the table. Cells that are not adjacent can be selected using Ctrl key when selecting. Selected items will be highlighted in black. You can subsequently paste the table into another application (such as a spreadsheet).

Read more: Selecting Items and Copying Values in Result Tables (page 567).

By default, Array results operate on Final Values. That is, the display uses the values at the end of each realization. However, by defining Capture Points, you can display values at any specified time. If you have created Capture Points, an additional drop-list is added to the display window to allow you to select the set of data (i.e., the values at the specified Capture Point) that you would like to display.

Read more: Viewing Results at Capture Points (page 569).

If you have run (and saved) multiple realizations, a realization control is added to the top of the display window for an Array result:

![Realization Control](image)

By default, the final (unscreened) realization is shown. You can type in a specific realization (and press Enter), move upward or downward through the available realizations using the spin control, or use the Ctrl-Up and Ctrl-Down keys to view the previous or next available realization, respectively.
Using Result Screening in Array Results

**Note:** You can only view those realizations which are available (i.e., those which have been saved and have not been screened out).

**Read more:** Using Result Screening in Array Results (page 699).

When carrying out probabilistic simulations, you may often run hundreds or thousands of realizations. In order to analyze the results, it is often quite useful to screen out some of the realizations by classifying the realizations into categories. A category is simply defined by a condition relating one or more outputs in the model (e.g., those realizations in which the discount rate was above 3.5%; those realizations in which the profit exceeded $1,000,000; those realizations in which the peak concentration was between 1 mg/l and 10 mg/l).

**Read more:** Classifying and Screening Realizations (page 577).

Classification categories are defined at the bottom of the Monte Carlo Result Display Properties dialog:

This dialog can be accessed from the Monte Carlo tab of the Simulation Settings dialog (via the Result Options... button), and is also accessible by pressing the Options... button available in the Result Properties dialog of an Array Result element.

You can screen a particular category from the results by clearing the **Include** box at the bottom of the Monte Carlo Result Display Properties dialog:

Only include those realizations in the categories which you have chosen to include can then be displayed in charts and tables.

**Note:** You can edit the **Include** box in Edit Mode, Result Mode Scenario Mode or Run Mode.
In the example above, the Low and High categories have been screened out, so when displaying results, only those realizations that fall into the Medium category would be available in the realizations spin control at the top of the display window:

![Array Results](image)

**Read more:** Viewing Multiple Realizations of Array Results (page 698).

GoldSim has powerful charting capabilities that allow you to customize the appearance of each chart. This includes adding headers and footers, and changing axis scales and labels for scatter plots. Most of these attributes can be edited by pressing the **Edit Chart Style** button at the top of the Multi-Variate display window:

![Chart Style](image)

Pressing this button (or right-clicking in a chart and selecting **Edit Chart Style**) provides access to the following dialog for editing the various chart properties:

![Chart Style Dialog](image)

This dialog is common to all types of charts, and is discussed elsewhere.

**Read more:** Editing the Appearance of a Chart (page 702).

For Vector Charts, the bar chart color is fixed and cannot be edited (solid red). You can, however, modify which items are actually included in the chart. Pressing the **Rows** button from the Result Properties dialog for the Array result displays the following dialog:
Viewing Array Results

Read more: Viewing the Properties of an Array Result (page 690).

Note: This dialog can also be accessed from the main GoldSim menu (by selecting Model | Array Labels... and then selecting the appropriate Array Label Set). When you access the dialog in this manner, the Style for each item is also displayed, but changing these has no impact on Vector Charts, since the Style is fixed.

By default, the Show box is checked for all items. If you clear this box, the row item will not be displayed in charts (or tables).

For Matrix Charts, the bar chart color for each column can be specified by pressing the Columns... button from the Result Properties dialog for the Array result. The following dialog will be displayed:

![Array Label Set dialog](image-url)
The Style column controls the style for the bars for the columns in the Matrix Chart. You can also modify which column items are actually included in the chart. By default, the Show box is checked for all items. If you clear this box, the column item will not be displayed in charts (or tables).

### Editing the Appearance of a Chart

Whenever you first view a result in GoldSim, it takes on a set of characteristics (e.g., fonts, axis titles, header) defined by a default chart style for that particular type of chart. In some cases, these default chart styles will be sufficient.

In other cases, however, you will want to customize the appearance of the chart. GoldSim allows you to do so. In addition, after customizing a set of properties for a particular type of chart, you can save this set as a new style which you can then selectively apply to subsequent charts (or set as the default for all subsequent charts of that type).

Selecting the Chart Style… button in any chart (or right-clicking on any chart and selecting Edit Chart Style from the context menu) displays the Chart Style dialog for editing the appearance of the chart:

![Chart Style Dialog](image)
This dialog allows you to edit the appearance of the chart (e.g., by modifying headers, footers, axis labels and scales, legends, etc.). To make it easier to modify and customize your charts, the same tabbed dialog is used for all chart types in GoldSim; some of the tabs and/or options, however, are hidden for those charts for which they are not applicable.

**Note:** When this dialog is displayed, the Chart display stays active. If you modify some part of the chart's appearance using the dialog and press the Apply button, the change is applied to the Chart. This allows you to view changes without closing the dialog.

The dialog has a number of tabs, which are described in the following sections.

The General tab is shown by default when you first open the Chart Style dialog.

**The Chart Style General Tab**

The General tab serves two purposes:

- The upper portion of the tab allows you to specify the colors and border styles for the chart; and
- The lower portion of the tab allows you to apply a specific chart style (a common group of appearance properties than can be applied to the entire chart), and (if changes have been made) allows you to overwrite the current chart style or save the current set of properties as a new chart style.

**Read more:** [Creating and Using Chart Styles](#) (page 712).

The upper portion of the General tab of the dialog is described below.

By default, the Chart Area defaults to the same colors as the Window Area and the Data Area defaults to the same colors as the Chart Area.

The Background Color controls the background.

The Foreground Color controls the text or lines that appear in the chart. The Window Area Foreground Color controls the header, the footer and the legend. The Chart Area Foreground Color controls the axes, the axis labels and the grid.
Colors are defined in a hierarchical manner for a chart by Window Area, Chart Area, and Data Area. The Window Area encompasses the Chart Area, and the Chart Area encompasses the Data Area, as illustrated below:

The Chart Style dialog for the example above (with three different background colors) would look like this:

You can place a border around the Window Area and/or the Chart Area of a specified style and width. You can also choose to display a bounding box around the Data Area by checking **Show axis bounding box**.

The Header and Footer tabs are used to specify a header and footer for the chart, and are identical in appearance.
If **Hide Header** (or **Hide Footer**) is checked, the header or footer is hidden. You can also hide or show the header or footer via the chart context menu (under View) or via the header/footer context menu.

You enter **Text** for the header or footer, and determine the **Justification** (if it consists of multiple lines), and **Font**.

You can place a border around the header or footer, and specify its **Style** and **Width** (in terms of pixels). You can also specify the **Foreground** and **Background** color for the header or footer (or default the colors so they use that of the Window Area, as specified in the General tab).

---

**Note:** You can access the Header or Footer tab of the chart style dialog directly by selecting **Edit Header…** or **Edit Footer…** from the header/footer context menu.

---

**Note:** If you right-click within the **Text** edit field, a context menu providing a list of keywords will be provided. Keywords allow you to automatically insert text that is determined automatically by the context of the result (e.g., the keyword `%Rname%` inserts the Result element name). This allows you to use a single chart style for multiple results.

---

**Read more:** [Using Keywords in Styles](page 717).

Depending on the type of chart and the number of outputs being plotted, up to three axis tabs may be present in the Chart Style dialog (there are always at least two):

<table>
<thead>
<tr>
<th>Chart Type</th>
<th>X-axis</th>
<th>Y-axis</th>
<th>Y2-axis</th>
<th>Z-axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time History</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Distribution</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The X, Y and Y2 axis tabs have the same appearance (with one minor exception noted below).

At the top of the dialog, GoldSim indicates what the particular axis represents, along with its units (if applicable).

The various fields in this tab are described below:

- **Title**: This is the axis title. If you right-click within the Title edit field, a context menu providing a list of keywords will be provided. Keywords allow you to automatically insert text that is determined automatically by the context of the result (e.g., the keywords %rlabel% %(unit)% inserts the result label and the display units for the output associate with the axis). This allows you to use a single chart style for multiple results.

  Read more: Using Keywords in Styles (page 717).

- **Title Rotation**: This is only available for the Y and Y2 axes. There are three options: 90 (the default), in which the title is rotated and read from bottom to top; 270, in which the title is rotated and read from top to bottom; and None, in which the title is not rotated (i.e., is horizontal), and is placed at the top of the axis.

- **Axis Font**: This provides access to a dialog for specifying the font for the axis. Note that this font is applied to all axes (i.e., all axes must use the same font).

- **Hide Axis**: If this box is checked the entire axis (and all annotation) is hidden.

- **Logarithmic**: If this box is checked, the axis is plotted logarithmically (rather than linearly). Note that if the data set you are plotting contains numbers which are less than or equal to zero, these points cannot be plotted on a logarithmic axis. Therefore, if you make the axis logarithmic in such a
situation, GoldSim will omit these points from the plot (i.e., they will appear as "holes" or "gaps" in the plot).

**Note:** Bar charts cannot be plotted on a logarithmic axis. For example, when plotting a distribution chart as a PDF, the X-axis cannot be logarithmic. If you check the **Logarithmic** box, it will be ignored. In addition, 2-D and 3-D scatter plots cannot be plotted using logarithmic axes. In these cases, the **Logarithmic** box is grayed out.

**Reversed:** If this box is checked, the axis numbering is reversed (e.g., going from high values to low values rather than from low values to high values).

**Label Placement:** This is only available for the X, Y and Y2 axes, and determines where the axis labels are placed relative to the other (orthogonal) axis. There are three options: On Axis (the default), in which the labels are placed alongside the axis, Minimum (in which they are placed at the minimum value of the other axis), or Maximum (in which they are placed at the maximum value of the other axis).

**Label Rotation:** This determines the angle that the axis numbers are rotated. The default is 0.

**Origin Placement:** This is only available for the X, Y and Y2 axes, and determines where the axis intersects the orthogonal axis (e.g., where the X-axis intersects Y-axis). There are three options: Minimum (the default), in which it intersects at the minimum value of the other axis; Maximum, in which it intersects at the maximum value of the other axis; or At Value, in which it intersects at the Value on the other axis indicated directly to the right.

**Note:** You cannot select label or origin placement for the Y axes for a time history chart with two vertical (Y and Y2) axes.

**Scale Minimum:** This is the minimum value (lower bound) on the axis. If **Auto** is checked, GoldSim will select the value for you (based on the range).

**Scale Maximum:** This is the maximum value (upper bound) on the axis. If **Auto** is checked, GoldSim will select the value for you (based on the range).

**Note:** If the axis being plotted is a date, the scale minimum and maximum will display date and time controls so you can specifically select a date/time range.

**Color:** This is the color of the axis and all of the annotation. If **Auto** is checked, the foreground color specified for the Chart Area (in the **General** tab) will be used.

**Axis Width:** This is the width of the axis lines, in pixels

**Tick Length:** This is the tick length, in pixels.
Note: One key attribute for charts is how values on axes are displayed. You cannot control the number of significant figures displayed (this is automatically determined). You can, however, control under what circumstances scientific notation is used via the Results tab of the Options dialog (accessed via Model | Options… from the main menu).

Read more: Controlling Significant Figures and Scientific Notation in Result Displays (page 568).

Note: When viewing a 3D scatter plot, some of the axis options are fixed by GoldSim and will automatically be grayed out in the X-Axis and Y-Axis tabs.

Specifying the Axis Appearance for 3D Charts

The Z-axis tab of the Chart Style dialog box is only available when viewing 3D scatter plots and 3D array plots and contains a subset of the options available on the other axes.

This is because the nature of a 3D plot is such that other than the title, the scale, the font and the font size, the appearance of the axis labels in a 3D plot are fixed by GoldSim and cannot be edited.

The various fields in this tab are described below:

**Title**: This is the axis title. If you right-click within the Title edit field, a context menu providing a list of keywords will be provided. Keywords allow you to automatically insert text that is determined automatically by the context of the result (e.g., the keywords %rlabel% %(unit)% inserts the result label and the display units for the output associate with the axis). This allows you to use a single chart style for multiple results.

Read more: Using Keywords in Styles (page 717).

**Axis Font**: This provides access to a dialog for specifying the font for the axis. Note that this font is applied to all axes (i.e., all axes must use the same font).
**Hide Axis**: If this box is checked the entire axis (and all annotation) is hidden.

**Scale Minimum**: This is the minimum value (lower bound) on the axis. If *Default* is checked, GoldSim will select the value for you (based on the range).

**Scale Maximum**: This is the maximum value (upper bound) on the axis. If *Default* is checked, GoldSim will select the value for you (based on the range).

**Grids (XY-Plane, XZ-Plane, YZ-Plane)**: This determines if grids are shown for the specified planes. If checked (and applicable to the plot), the grids are shown.

**Show Drop Lines**: If this box is checked, GoldSim will draw a line from the data point to the XY-plane for scatter plots. It does not apply to array charts.

The Legend tab of the Chart Style dialog is used to specify the legend for the chart, and is only available for those charts for which a legend is applicable.

If *Hide Legend* is checked, the legend is hidden. You can also hide or show the legend via the chart context menu (under *View*) or via the legend context menu.

You enter the *Title* for the legend, and determine its *Font*. If the *Reverse Order* box is checked, the items of the legend are listed in reverse order.

The *Position* field allows you to place the legend at various locations around the chart (Top, Bottom, Right, Left, etc.). The *Orientation* of the items in the legend can be specified as being vertical or horizontal.

You can place a border around the legend, and specify its *Style* and *Width* (in terms of pixels). You can also specify the *Foreground* and *Background* color for the legend (or default the colors by selecting *Auto* so they use that of the Window Area, as specified in the *General* tab).

The Grid tab of the Chart Style dialog is used to specify the grid for the chart.
Editing the Appearance of a Chart

Note that the grid lines are specified for each axis (and are perpendicular to the axis).

You can specify the **Line Style** and **Line Width** (in terms of pixels). You can also specify the **Color** for the grid lines.

You specify the **Spacing** of the grid lines in terms of the actual units for the data assigned to the axis.

If **Auto** is checked, GoldSim automatically computes an appropriate spacing based on the data range, defaults the color so it uses the Chart Area foreground color (as specified in the **General** tab), and resets the **Line Style** and **Line Width** to default settings.

Unlike the other attributes of a chart (e.g., axes labels, headers, legends), the manner in which the data itself is shown in the chart is not a property of the chart (and hence is not edited in the Chart Style dialog), but is determined by the type of result and the context in which it is being plotted (e.g., is the result an array, are scenarios being viewed, have classification categories been defined).

As such, Data Styles can be specified in multiple places within GoldSim and the style that is used for a particular chart is a function of the context for the display (discussed below).

In all cases, however, the Data Style dialog for a particular data set looks like this:

---

**Editing Data Styles**
For each data set, you can select the **Line Style** (e.g., solid, dashed, etc.), the **Line Width** (in pixels) and the **Line Color**. You can also choose to plot a symbol at the data points (the default is no symbol is shown), and select a **Symbol Style** (e.g., square, solid circle, etc.), **Symbol Size** (in pixels), and **Symbol Color**. For bar charts, you can also choose the **Fill Style**. The selected line color is used as the fill color.

There are several different places from where this dialog can be accessed. It is easiest to summarize this by examining each type of result separately.

**Time History Results**

- In most situations, the Data Style is specified directly in the Result Properties dialog.

  *Read more:* [Viewing the Properties of a Time History Result](page 581).

- If you are plotting an array, the Data Style for the different array items is defined in the Array Label Set dialog.

  *Read more:* [Viewing Time Histories for Array Outputs](page 592).

- If you have run multiple realizations and you are displaying **Probabilities**, the Data Style for the display is specified at the top of the Monte Carlo Result Display Properties dialog (in the section labeled “History Statistics”). This dialog is accessed by pressing the **Options…** button in the Result Properties dialog of a Time History Result element.

  *Read more:* [Viewing Probability Histories for Multiple Realizations](page 600).

- If you have run multiple realizations; 2) you have defined more than one category; and 3) you are displaying **All Realizations**, GoldSim will label the curves in the Time History Chart based on the categories you have defined. The Data Style for categories is defined at the bottom of the Monte Carlo Result Display Properties dialog (in the section labeled “Realization Classification and Screening”). This dialog is accessed by pressing the **Options…** button in the Result Properties dialog of a Time History Result element.

  *Read more:* [Using Result Classification and Screening in Time History Results](page 615).
• When a model is in Scenario Mode, the Data Style for the different scenarios is controlled by the Scenario Manager dialog (accessed from the main menu (Run | Scenario Manager...) or by pressing F7).

Read more: Viewing Scenario Results in Time History Result Elements (page 622).

Distribution Results

• In most situations, the Data Style is specified directly in the Result Properties dialog.

Read more: Viewing the Properties of a Distribution Result (page 640).

• If you have run multiple realizations; 2) you have defined more than one category; and 3) the Distribution Result has only a single result defined, GoldSim will label the curves in the Distribution Chart based on the categories you have defined. The Data Style for categories is defined at the bottom of the Monte Carlo Result Display Properties dialog (in the section labeled “Realization Classification and Screening”). This dialog is accessed by pressing the Options... button in the Result Properties dialog of a Time History Result element.

Read more: Using Result Classification and Screening in Distribution Results (page 658).

• When a model is in Scenario Mode, the Data Style for the different scenarios is controlled by the Scenario Manager dialog (accessed from the main menu (Run | Scenario Manager...) or by pressing F7).

Read more: Viewing Scenario Results in Distribution Result Elements (page 665).

Multivariate Results

• In the absence of Classification categories, the symbol used for a scatter plot is fixed and cannot be edited (a solid red square).

• If you defined more than one category, GoldSim will label the points in the Scatter Plots based on the categories you have defined. The Data Style for categories is defined at the bottom of the Monte Carlo Result Display Properties dialog (in the section labeled “Realization Classification and Screening”). This dialog is accessed by pressing the Options... button in the Result Properties dialog of a Time History Result element.

Read more: Using Result Classification and Screening in Multi-Variate Results (page 686).

Array Results

• The Data Style for the different array items is defined in the Array Label Set dialog.

Read more: Controlling the Chart Style in Array Results (page 700).

Creating and Using Chart Styles

Often, after modifying the appearance of a chart, it is very useful to save the particular set of properties you have created (e.g., label fonts, header, footer) so that you can apply them to another chart. To facilitate this, GoldSim allows you
to save the set of properties as a named chart style. A chart style consists of all of the chart properties that you specify in the Chart Style dialog.

Read more: Editing the Appearance of a Chart (page 702).

The named chart style is saved as part of the model file, and can subsequently be applied to other results in your model. You can also export and import chart styles between models.

Note: Unlike the other attributes of a chart (e.g., axes labels, headers, legends), the manner in which the data itself is shown in the chart is not a property of the chart itself (and hence is not edited in the Chart Style dialog and is not saved in a chart style.)

Read more: Editing Data Styles (page 710).

Whenever you view a result, it will initially take on a set of characteristics defined by the default chart style for that particular type of chart (e.g., time history, distribution, etc.).

GoldSim provides four "built-in" chart styles:

- Basic Time History Style
- Basic Distribution Result Style
- Basic Multi-Variate Style
- Basic Array View Style

Initially, the four "built-in" styles are defined as the default chart styles for each type of plot. For any type of chart, you can specify one of your own custom chart styles as the default chart style for all subsequent results that are displayed.

After you have customized the appearance of a particular result, you can save a new chart style as follows:

1. From the General tab of the Chart Style dialog, press the Save Style… button.
2. To save a style, press the Save Style… button. The following dialog is displayed:

![Save Style Dialog](image)

3. Type in a style name (to create a new style) or select an existing style from the drop list (to overwrite an existing style).
4. Press OK to save the style.

Note: You cannot overwrite the "built-in" styles provided by GoldSim. As a result, the drop list only contains user-defined styles.

You can apply a chart style to a result chart as follows:
Creating and Using Chart Styles

1. From the General tab of the chart editing dialog, press the Apply Style… button.

2. The following dialog is displayed:

   ![Apply Style Dialog](image)

   Select the style you wish to apply, and press OK.

3. Select the style you wish to apply, and press OK.

   **Note:** Styles store all the information required for all types of charts (with the exception of the Data Style). Hence, any style can be applied to any kind of chart. In practice, however, you will typically create and apply different styles for different types of charts in order to better customize their appearance.

Using the Style Manager

The GoldSim Style Manager is used to manage all of your styles. You can use the Style Manager to view where styles are being used, create new styles, import and export styles, and to define default styles.

The Style Manager is accessed from the main menu via **View|Style Manager**:

![Style Manager](image)

The Style Manager consists of two tabs: Manage and Options. The Manage tab lists all of the styles in your model (including the "built-in" styles provided by GoldSim). When you select a style, a preview (using a simple default data set), showing its primary attributes is shown in the Preview window. You can also enter a Description for any user-defined style (in order to remind yourself of the purpose or contents of the style).

You can delete a style by selecting it and pressing the Remove button. Note, however, that this button is grayed out if a "built-in" style is selected ("built-in" styles cannot be deleted).
If you right-click on one of the listed styles, a list of the Result elements which use that style is displayed.

*Read more:* Creating and Using Result Elements (page 570).

You can use the Style Manager to copy the properties from one style to another, as well as import and export styles.

In some situations, you may have applied a style to a number of Result elements, and then want to edit that style so it is identical to some other style in your model. Rather than editing the style, you can simply copy all the properties of an existing style into another style as follows:

1. Press the **Copy from...** button from the **Edit Style** tab of the Style Manager. The following dialog is displayed:

   ![Copy from dialog](image)

2. Select an existing style from the **Copy from** drop-down list.

3. Press **OK**.

Chart styles are saved within the model file. Often, however, you may wish to use a style that you created in one model file within a second model file. To facilitate this, GoldSim allows you to save a style as an external *style file*, and then create a new style (within a different model file) by importing the external style file.

You can export a style (i.e., create a style file) as follows:

1. Press the **Export...** button from the **Manage** tab of the Style Manager. A Save As dialog will be displayed for saving the style to a file. The default filename is identical to that of the style. If you wish, you can modify it. Style files have the extension ".gcs".

2. Press the **Save** button to save the style file.

   **Note:** The default location in which the style file will be saved is specified in the **Options** tab of the Style Manager. If desired, of course, you can also save the style file to a different folder.

You can import a style file into a model file as follows:

1. Make sure that the style file that you wish to import is in the default style file folder (specified in the **Options** tab of the Style Manager).

2. Press the **New...** button from the **Manage** tab of the Style Manager. The following dialog will be displayed:

   ![New Style dialog](image)
3. If one of the built-in styles was selected when your pressed the **New** button, you can enter a **Style Name** for the style into which you wish to import the information contained in the style file (and a new style will be created). Otherwise, it will be imported into the selected style (overwriting it).

4. Expand the **Based on** drop list to display all of the available styles upon which you can base the new style.

5. In addition to including all of the styles, the list also includes all the style files located in the default style file directory. (Style files include the date they were created.)

6. Select the style file that you wish to import and press **OK**.

The new style will be added to the style list. You may want to edit the **Description** for the new file to indicate how it should be used.

Whenever you view a result, it will initially take on a set of characteristics defined by the default chart style for that particular type of chart (e.g., time history, distribution, etc.).

GoldSim provides four "built-in" chart styles:

- Basic Time History Style
- Basic Distribution Result Style
- Basic Multi-Variate Style
- Basic Array View Style

Initially, these four "built-in" styles are defined as the default chart styles for each type of plot. For any type of chart, however, you can specify one of your own custom chart styles as the default chart style for all subsequent results that are displayed.

This is done from the **Options** tab of the Style Manager. The default styles for each of the four types of result charts are specified at the top of this tab of the dialog.
Creating and Using Chart Styles

Note: Styles store all the information (except for Data Styles) required for all types of charts. Hence, any style can be applied to any kind of chart. In practice, however, you will typically create and apply different styles for different types of charts in order to better customize their appearance. Hence, it is likely that you will want to assign a different default chart style for each type of chart.

Using Keywords in Styles

In order to facilitate the use of a single chart style for multiple results, GoldSim allows you to use keywords (delimited by the % symbol) when you create and use chart styles. Keywords can be used in fields where titles or text is required (e.g., headers and footers, axis labels, legend titles). For example, the keyword %rname% inserts the name of the Result element being displayed, and the keyword %#real% inserts the number of realizations. We could insert these in a header, as shown below:

The keyword %rname% is actually the default header text for all results. It is hidden for interactive results (since there is no corresponding Result element).

This would appear in the chart as follows:
You can access the full list of keywords which are available by right-clicking in any Title or Text field in the chart style dialog.

A complete list of available keywords is provided below:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%rname%</td>
<td>Result element name[^1]</td>
</tr>
<tr>
<td>%rdesc%</td>
<td>Result element description[^1]</td>
</tr>
<tr>
<td>%rlabel%</td>
<td>Label of the first result mapped to the axis[^2] (context sensitive)</td>
</tr>
<tr>
<td>%unit%</td>
<td>Display units for axis value[^3] (context sensitive)</td>
</tr>
<tr>
<td>(unit)%</td>
<td>Display units for axis value within parentheses[^3] (context sensitive)</td>
</tr>
<tr>
<td>[%unit]%</td>
<td>Display units for axis value within brackets[^3] (context sensitive)</td>
</tr>
<tr>
<td>%#real%</td>
<td>Number of realizations</td>
</tr>
<tr>
<td>%#creal%</td>
<td>Number of completed realizations[^4]</td>
</tr>
<tr>
<td>%mctype%</td>
<td>Type of Monte Carlo simulation (Deterministic or Probabilistic)</td>
</tr>
<tr>
<td>%rundate%</td>
<td>Date simulation was started (displayed using Regional date format settings)</td>
</tr>
<tr>
<td>%runtime%</td>
<td>Time simulation was started (displayed using Regional time format settings)</td>
</tr>
<tr>
<td>%runduration%</td>
<td>Duration of simulation (simulation run time)</td>
</tr>
<tr>
<td>%curdate%</td>
<td>Current date (displayed using Regional date format settings)</td>
</tr>
<tr>
<td>%curtime%</td>
<td>Current time (displayed using Regional time format settings)</td>
</tr>
<tr>
<td>%period%</td>
<td>Period Label of reporting period being displayed (blank if result is not period-based)</td>
</tr>
<tr>
<td>%ctprec%</td>
<td>CT Module solution precision (Low, Medium, High). N/A shown if module not being used.</td>
</tr>
<tr>
<td>%author%</td>
<td>Model Author (as specified in the Information tab of the Simulation Settings dialog)</td>
</tr>
<tr>
<td>%an_desc%</td>
<td>Analysis Description (as specified in the Information tab of the Simulation Settings dialog)</td>
</tr>
</tbody>
</table>
Exporting Results

In some cases, rather than using the result plotting and post-processing capabilities provided by GoldSim, you may wish to plot, analyze, or store the results using a separate program, such as a spreadsheet, a database, or statistical analysis package.

To facilitate this, GoldSim provides three flexible methods for exporting results in the form of an MS-Excel file or a text file. The three methods are as follows:

Exporting Time Histories from a Time History Result Element. You can export outputs that are specified within a Time History Result to an MS-Excel spreadsheet file or a text file. This export can be set to occur automatically at the end of a simulation, or can be triggered manually while in Result or Scenario Mode. Spreadsheet export provides export of a single history per result (for single realization runs) or a specified statistic (for multi-realization runs). Text export provides export of all realizations for multi-realization runs.

Exporting Results Using a Spreadsheet Element. You can export specified GoldSim outputs to an MS-Excel spreadsheet file using a Spreadsheet element. Any GoldSim output can be exported in this way. By using the built-in capabilities of Spreadsheet elements (in particular, the ability to specify dynamic offsets), time histories and/or multiple realizations can be output. This export occurs automatically during the simulation.

Exporting Final Value Results for Multiple Outputs and Multiple Realizations. You can export the final values for each realization for specified GoldSim outputs to a text file. The export is carried out manually using a key combination when viewing a Raw Multi-Variate Data Table. This export is of particular value when you wish to carry out sensitivity and/or uncertainty analyses using a separate statistical analysis software package.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%filename%</td>
<td>model filename (without extension)</td>
</tr>
<tr>
<td>%filename_ex%</td>
<td>model filename (with extension)</td>
</tr>
<tr>
<td>%copyright%</td>
<td>GoldSim copyright notice</td>
</tr>
<tr>
<td>%version%</td>
<td>GoldSim version number</td>
</tr>
</tbody>
</table>

1 If an interactive result is being displayed, this is ignored.

2 If the keyword is used in a location other than an axis title (e.g., a header or footer title), it simply appears as entered (e.g., %rlabel%) and is not replaced with a result label (as it would ambiguous as to which label to use).

3 If the axis data is dimensionless, or if it represents a date/time, the keyword is ignored. If the keyword is used in a location other than an axis title (e.g., a header or footer title), it simply appears as entered (e.g., %unit%) and is not replaced with a unit (as it would ambiguous as to which unit to use).

4 Displays total number of completed realizations, even if some are currently screened. In Scenario Mode, if all of the displayed scenarios do not have the same number of completed realizations, N/A is displayed.

Note that some of these keywords are used as defaults. For example, all result displays use %rname% in the Header, and the axis titles use both %rlabel% and %unit%.

Exporting Results
These methods are described in detail in the sections below.

You can export outputs that are specified within a Time History Result to an MS-Excel spreadsheet file. This export can be set to occur automatically at the end of a simulation, or can be triggered manually while in Result or Scenario Mode. In addition, when running multiple scenarios, scenario comparisons can be written to the spreadsheet.

Read more: Exporting Scenario Results from a Time History Result Element to a Spreadsheet (page 725).

To enable this capability for a Time History Result element, select “MS-Excel” from the Export Results To drop-list at the bottom of the Time History Result element dialog:

By default, this is set to “None”. When you select “MS-Excel”, an Export tab is added to the dialog:

You must first specify the Excel filename to which you wish to export. You can enter the name of a Microsoft Excel spreadsheet file to which you wish to link by pressing the Options >> button. This will provide options for either selecting an existing file, or creating (and then selecting) a new file. You can also type in the name of a file directly. In this case, you must provide a full or relative path (relative to the model file).

The Excel file does not need to exist prior to the export. If you type in a file that does not exist, GoldSim will create it at the time of export. The filename that you enter can contain one or more of the following keywords which will be replaced with the appropriate text at the time of export:
Exporting Results

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%en%</td>
<td>The name of the Result element</td>
</tr>
<tr>
<td>%filename%</td>
<td>The name of the GoldSim file (without the extension)</td>
</tr>
<tr>
<td>%rundate%</td>
<td>Date simulation was started (displayed using Regional time format settings)</td>
</tr>
<tr>
<td>%runtime%</td>
<td>Time simulation was started (displayed using Regional time format settings).</td>
</tr>
</tbody>
</table>

If the file exists at the time of export, it will be overwritten by the new data. If **Confirm before overwriting contents of an existing file** is checked, GoldSim will confirm before doing so. Note, however, that for existing spreadsheet files, GoldSim only overwrites those cells to which it is specifically exporting results. It does not overwrite any other cells in the file. In some cases, this could lead to situations where only a portion of the data in a particular row or column is overwritten (e.g., if the number of timesteps is changed between simulations), and this has the potential to create confusion. To prevent this, you should check **Clear previously written cell range before export**. If this is checked, GoldSim clears out all the cells from the previous time it exported from this Result element before writing the new results.

**Note**: In order to be able to clear cells that were previously written to, GoldSim must “remember” the cells to which it has previously written by saving this information with the file after the export has taken place. As a result, if you carry out an export, and then close the file without first saving it, this information is lost (i.e., GoldSim does not “remember” the cells to which it has previously written). Hence, to take advantage of this feature, you should always save the model after you carry out an export.

In addition to specifying the Excel filename, you must also specify the **Name of destination worksheet and starting cell** (the sheet and starting cell) where the data is to be exported. If the Excel file already exists, you can use the **Location…** button, which opens the spreadsheet and provides a dialog allowing you to directly select the desired cell using your mouse. If the Excel file does not exist, you must manually enter the sheet name and starting cell.

**Note**: Scenario results are exported to a different sheet. In particular, the name of the sheet is “Sheetname-Scenarios”, where Sheetname is the name that you specify in this dialog.

**Read more**: Exporting Scenario Results from a Time History Result Element to a Spreadsheet (page 725).

Data is always exported in columns (one column for the time, and one column for each result). If only a single realization is run, that realization is exported (for each result specified in the Result element).

If multiple realizations are run, by default the specified **Custom Statistic** for each result is exported. However, if you check **For multi-realization Monte Carlo simulations export History Statistics instead of Custom Statistic**, then multiple statistics (those displayed in Probability History displays, and defined in the Monte Carlo Result Options dialog) will be exported.
Read more: Viewing Custom Statistics for Multiple Realizations (page 603); Viewing Probability Histories for Multiple Realizations (page 600).

Note: If you need to export all realizations of a multiple realization simulation, you can do so by choosing to export to a text file rather than a spreadsheet.

Read more: Exporting from a Time History Result Element to a Text File (page 727).

An example of what the exported data in a spreadsheet looks like is provided below:

In this particular example, the Result element contained two results (Height and Flow_Rate). The model was run for multiple realizations, and the default choice of exporting the Custom Statistic was selected (which in this case was Mean for Height and the 50\textsuperscript{th} percentile for Flow_Rate).

Note that GoldSim writes three header rows. The first identifies the result, the second the units, and the third the result type (in this example, the Mean and the 50\textsuperscript{th} percentile). In addition, the first cell contains a comment with information regarding the run (e.g., version used, date/time of simulation and export, name of Result element, etc.).

Note: GoldSim exports the results as single-precision values. When exporting dates from a calendar time model, if the first cell of the time column is not formatted as a date, GoldSim will format the column as a localized date (based on Windows settings). None of other columns (i.e., the result values) will be formatted.

The time points that are exported to the spreadsheet are the same as those displayed in result charts and tables for the result element.

Read more: Specifying When Time History Results Will Be Saved (page 464).

If your Result element is configured to display Reporting Periods, and you have more than one Reporting Period defined, you must specify which period you wish to export in the When using Reporting Periods drop-list. The type of Reporting Period results that are exported (e.g., Average, Cumulative, etc.) are as specified in the Properties dialog for the Result element.
If a result is an array, it is simply treated in the spreadsheet as separate items (i.e., an array of 10 items would produce 10 result columns).

Several other points regarding result export to spreadsheets should be noted:

- Conditions are exported as 0 (False) or 1 (True).
- Results that are dates (such as the output of a Milestone element) are exported as Julian days.
- If an output is an array, only those items of the array which have been selected to be viewed (via the Array Label Set dialog) are exported.

**Read more:** Viewing Reporting-Period Based Results in Time History Result Elements (page 607).

- For simulations involving multiple realizations, only those realizations which have been specifically selected to be included (via the Monte Carlo Result Options dialog) are used to compute the Custom Statistic that is exported.

**Read more:** Classifying and Screening Realizations (page 577).

- High resolution results are not exported. That is, if your Time History Result is configured to include unscheduled updates, these are not exported. Only scheduled updates are exported.

**Read more:** Viewing Unscheduled Updates in Time History Result Elements (page 629).

- You cannot export time history results from a SubModel (i.e., if the Time Display Setting in the Result element is “SubModel Time”).

The actual export of the results can be triggered in two ways: manually and automatically. This is determined by the **Export automatically when simulation completes** checkbox. If this box is cleared, GoldSim only exports the data if and when you press the **Export Now** button in the Export tab of the Result element dialog. If this box is checked, however, GoldSim automatically exports the results at the end of the simulation. When it exports the results in this way, any errors it encounters (e.g., running out of space in the spreadsheet) are written to the Run Log.

For elements which are set to export automatically, you can globally control their behavior from the **Results** tab of the Options dialog (accessed from the main menu via **Model|Options...**, and then selecting the **Results** tab):
The options in the drop-list affect all Time History Result elements which are set to export automatically when the simulation completes. If “Export results after simulation” is selected, any Result elements which are set to export results will export (silently) at the end of the simulation. If “Prompt before exporting results” is selected, at the end of the simulation, you will be prompted to determine whether or not to carry out an automatic export. If “Do not export results” is selected, no automatic export is carried out (i.e., this overrides the selection for each individual Result element).

Pressing the Export Now button manually exports results from all Time History Result elements.

**Note:** Within Dashboards, the Button control provides a command option that is equivalent to pressing the Export Now button from the Result tab of the Options dialog.

Button command options within a Dashboard are discussed in detail in the Dashboard Authoring Module User’s Guide.

Several points should be noted regarding automatic export of spreadsheet results:

- Automatic export is ignored when carrying out a Sensitivity Analysis or an Optimization run.
  
  **Read more:** Running Sensitivity Analyses (page 539); Running an Optimization (page 528).

- Automatic export is ignored if the Result element is inside a SubModel.
  
  **Read more:** Using SubModels to Embed Models Within Models (page 973).

- Automatic export is only carried out under special circumstances when running and comparing scenarios.
Finally, when exporting to spreadsheets, the following points regarding the use of Excel should be noted:

- GoldSim supports .xlsx, .xlsm, and .xls Excel files. However, if you have an older version of Excel (prior to Office 2007), you will need to install Microsoft’s Office Compatibility Pack in order to read .xlsx and .xlsm files. Note that GoldSim does not officially support versions of Excel prior to Excel 2003.

- The .xls format is limited to 65,536 rows and 256 columns. Exporting in columns allows a maximum of 65,533 plot points to be exported (three rows are reserved for the header) and 255 output values (first column is reserved for time). If the data will not fit in the spreadsheet due to one of these limitations, a warning message will be written to the screen (if exporting manually) or the Run Log (if exporting automatically). Note, however, that Excel 2007 and later support an extended worksheet size (1,048,576 rows by 16,384 columns). If you wish to export data to an extended worksheet range into GoldSim, you must use Excel 2007 or newer, and the file format must be .xlsx or .xlsm.

- You should never try to open a spreadsheet that is linked to GoldSim while a simulation is running. In addition, prior to running a GoldSim model that is linked to a spreadsheet, all referenced spreadsheets should be closed. If you try to interact with a spreadsheet that is being used by GoldSim during a simulation, all interaction with the spreadsheet is blocked by GoldSim. However, under some circumstances this may not be possible and could lead to errors. Hence, as a general rule, you should not use Excel while a GoldSim model that references Excel is running.

When running multiple scenarios, time histories for different scenarios can be written to a spreadsheet.

Read more: Creating, Running and Comparing Scenarios (page 504).

Note, however, that export of scenario results to a spreadsheet must be carried out in a special way to prevent each export from simply overwriting previously exported scenarios.

In particular, when exporting scenarios, GoldSim exports all scenarios with results to a separate sheet from that defined in the dialog; it appends “-Scenarios” to the specified sheet name, and exports results to that sheet (creating it if it does not already exist). For example, if your specified sheet name was simply “Sheet1”, scenario results would be exported to “Sheet1-Scenarios” (and this is indicated in the dialog):

- Scenario results are exported only under the following four conditions:
1. If the model is in Scenario Mode, pressing **Export Now** from the **Export** tab of the Result element exports results for all scenarios with results to the specified Scenarios sheet.

2. If the model is in Scenario Mode, pressing **Export Now** from the **Results** tab of the Model Options dialog exports results for all scenarios with results from **all** Time History Result elements to their specified Scenarios sheet.

3. Within Dashboards, the Button control provides a command option that is equivalent to pressing the **Export Now** button from the **Result** tab of the Options dialog. Hence, if the model is in Scenario Mode, this button command exports results for all scenarios with results from all Time History Result elements to their specified Scenarios sheet. Button command options within a Dashboard are discussed in detail in the **Dashboard Authoring Module User's Guide**.

4. If you press **Run All** in the Scenario Manager, any Result elements that are configured to automatically export results will export results for all scenarios to their specified Scenarios sheet.

Several important points should be noted regarding scenario export:

- If you run a single scenario from the Scenario Manager or from the Scenario control on a Dashboard, no results are exported (even for Result elements that are configured to automatically export).

- If you run a model such that it ends in Result Mode (e.g., by pressing **F5** or the **Run** button in the Run Control toolbar), any Result elements that are configured to automatically export results will simply export results for the Active Scenario to the selected primary sheet (e.g., Sheet1). Similarly, if you press **Export Now** in Result Mode, the results for the Active Scenario will be exported to the primary sheet. In either case, any other results in the primary sheet will be overwritten. As a result, **you cannot export scenario comparisons in this way**.

- Since Player users cannot access the Scenario Manager, and single scenario runs are never automatically exported, if a Scenario control is present on a Dashboard when running the Player, automatic export will always be ignored. Results can only be exported manually. To export scenarios from the Player, the author should provide a Button control with an Export command (item #3 above). Note that the Result tab of the options dialog is accessible from the Player, so if the model is browsable, scenario export could also be carried out by pressing the **Export Now** button in that dialog (item #2). However, this would typically be a very awkward way for a Player user to trigger an export. The Scenario control within a Dashboard is discussed in detail in the **Dashboard Authoring Module User's Guide**.

When exporting scenario results to the Scenarios sheet GoldSim provides two options for how the results are exported on the **Export** tab of the Result element dialog:
Exporting Results

If you select “Group Results Together”, if multiple results are being exported, the columns will be grouped together by result:

In this particular example, the Result element contained two results (Flow and Cost) and three scenarios (Scenario 1, Scenario 2, and Scenario 3) and was run for a single realization. Note that all the results for Flow are presented (for each scenario), followed by all the results for Cost.

If you select “Group Scenarios Together”, if multiple results are being exported, the columns will be grouped together by scenario:

Note that in this case all the results for the first scenario are presented (for each result), followed by all the results for the second scenario, and finally all the results for the third scenario.

Note that when exporting scenario results, GoldSim writes four header rows (rather than three). The first identifies the scenario, the second the result, the third the units, and the fourth the result type. In this example, in which only a single realization was run, the result type is simply the realization number. If multiple realizations were run, the result type would be a statistic.

In addition, the first cell contains a comment with information regarding the run (e.g., version used, date/time of simulation and export, name of Result element, names of scenarios being exported, etc.).

You can export outputs that are specified within a Time History Result to a tab-delimited text file. This export can be set to occur automatically at the end of a simulation, or can be triggered manually while in Result Mode. Unlike spreadsheet exports, when running multiple realizations, instead of outputting Exporting from a Time History Result Element to a Text File
only a single statistic for each result, in addition, all realizations can be exported to the text file.

**Note:** You cannot export scenario results to a text file. If you need to export scenario results, you can do so by choosing to export to a spreadsheet rather than a text file.

*Read more:* Exporting Scenario Results from a Time History Result Element to a Spreadsheet (page 725).

To enable this capability for a Time History Result element, select “Text File” from the **Export Results To** drop-list at the bottom of the Time History Result element dialog:

By default, this is set to “None”. When you select “Text File”, an Export tab is added to the dialog:

You must first specify the text filename to which you wish to export. You can enter the name of an existing file to which you wish to link by pressing the **Options >>** button. This will provide options for either selecting an existing text file (with the extension .txt), or creating (and then selecting) a new file. You can also type in the name of a file directly. In this case, you must provide a full or relative path (relative to the model file).

**Note:** The output file must have the extension “.txt”.
The file does not need to exist prior to the export. If you type in a file that does not exist, GoldSim will create it at the time of export. The filename that you enter can contain one or more of the following keywords which will be replaced with the appropriate text at the time of export:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%en%</td>
<td>The name of the Result element</td>
</tr>
<tr>
<td>%filename%</td>
<td>The name of the GoldSim file (without the extension)</td>
</tr>
<tr>
<td>%rundate%</td>
<td>Date simulation was started (displayed using Regional time format settings)</td>
</tr>
<tr>
<td>%runtime%</td>
<td>Time simulation was started (displayed using Regional time format settings)</td>
</tr>
</tbody>
</table>

If the file exists at the time of export, it will be overwritten by the new data. If Confirm before overwriting contents of an existing file is checked, GoldSim will confirm before doing so.

You then must specify what you would like to export. If running a single realization, the options are not relevant. However, when running multiple realizations, you can export two kinds of information:

- You can export the individual histories for each result and/or you can export the Custom Statistic for each result.

Note: If you wish to export multiple statistics, you can do so by simply adding multiple result items to the Result element that reference the same output but have different Custom Statistics. Alternatively, you could choose to export to a spreadsheet, which provides an option for export multiple statistics (i.e., those used for Probability Histories).

Read more: Exporting from a Time History Result Element to a Spreadsheet (page 720).

Data is always exported in a tab-delimited format, with header information written at the top of the file. If multiple realizations are run, all (unscreened) realizations are written to the file. Multiple results are exported in separate blocks (unless the results reference the same output, have the same label and the same axis mapping, in which case they are exported in the same block so as to list all results referencing the same output together).

An example of what the exported data in a text file looks like is provided below:
In this particular example, the Result element contained two results (Flow and Cost). The model was run for 5 realizations, and contains 5 timesteps. If you export both realizations and one or more statistics, for each result, the realizations are listed first, followed by columns for the statistics.

Note that the first 10 lines contain header information (each preceded by a “!”):

- Version number
- Filename
- Time simulation was run
- Time export took place
- Name of Result element
- Full path to Result element
- Description for Result element
- The number of time points
- The number of realizations
- The relative realization weights (only included if realization are exported)

The time points that are exported to the text file are the same as those displayed in result charts and tables for the result element.

Read more: Specifying When Time History Results Will Be Saved (page 464).

The relative realization weights are normally all 1. However, if you use importance sampling (e.g., for Stochastics or Timed Events) or directly specify weights in the Simulation Settings dialog, they will be different and it is critical to take this into account for any post-processing that you carry out on results.

Read more: Applying Importance Sampling to a Stochastic Element (page 181); Timed Event Elements (page 365); Probabilistic Simulation Options (page 479).

After the header information, the result data is written. For each result, there are two header rows (preceded by a “!”) indentifying the result name and the units. The realization numbers are then shown, followed by a row for each timepoint. Each row contains all the (unscreened) realizations.

Note: GoldSim exports the results as single-precision values.
If your Result element is configured to display Reporting Periods, and you have more than one Reporting Period defined, you must specify which period you wish to export in the **When using Reporting Periods** drop-list. The type of Reporting Period results that are exported (e.g., Average, Cumulative, etc.) are as specified in the Properties dialog for the Result element.

**Read more:** [Viewing Reporting-Period Based Results in Time History Result Elements](page 607).

If a result is an array, it is simply treated in the text file as separate items (i.e., an array of 10 items would produce 10 result groupings).

Several other points regarding result export to text files should be noted:

- Conditions are exported as 0 (False) or 1 (True).
- All dates are exported wrapped in quotes.
- If an output is an array, only those items of the array which have been selected to be viewed (via the Array Label Set dialog) are exported.

**Read more:** [Viewing Time Histories for Array Outputs](page 592).

- For simulations involving multiple realizations, only those realizations which have been specifically selected to be included (via the Monte Carlo Result Options dialog) are exported.

**Read more:** [Classifying and Screening Realizations](page 577).

- High resolution results are not exported. That is, if your Time History Result is configured to include unscheduled updates, these are not exported. Only scheduled updates are exported.

**Read more:** [Viewing Unscheduled Updates in Time History Result Elements](page 629).

- You cannot export time history results from a SubModel (i.e., if the **Time Display Setting** in the Result element is “SubModel Time”).

The actual export of the results can be triggered in two ways: manually and automatically. This is determined by the **Export automatically when simulation completes** checkbox. If this box is cleared, GoldSim only exports the data if and when you press the **Export Now** button in the Export tab of the Result element dialog. If this box is checked, however, GoldSim automatically exports the results at the end of the simulation.

For elements which are set to export automatically, you can globally control their behavior from the **Results** tab of the Options dialog (accessed from the main menu via Model|Options..., and then selecting the **Results** tab):
The options in the drop-list affect all Time History Result elements which are set to export automatically when the simulation completes. If “Export results after simulation” is selected, any Result elements which are set to export results will export (silently) at the end of the simulation. If “Prompt before exporting results” is selected, at the end of the simulation, you will be prompted to determine whether or not to carry out an automatic export. If “Do not export results” is selected, no automatic export is carried out (i.e., this overrides the selection for each individual Result element).

Pressing the Export Now button manually exports results from all Time History Result elements.

**Note:** Within Dashboards, the Button control provides a command option that is equivalent to pressing the Export Now button from the Result tab of the Options dialog.

Button command options within a Dashboard are discussed in detail in the Dashboard Authoring Module User’s Guide.

Several points should be noted regarding automatic export of text file results:

- Automatic export is ignored when carrying out a Sensitivity Analysis or an Optimization run.

  **Read more:** Running Sensitivity Analyses (page 539); Running an Optimization (page 528).

- Automatic export is ignored if the Result element is inside a SubModel.

  **Read more:** Using SubModels to Embed Models Within Models (page 973).

- Automatic export is ignored when running and comparing scenarios.

An additional method that can be used to export results from GoldSim is to use a Spreadsheet element. A spreadsheet element can be used to export any output to a spreadsheet. By using the built-in capabilities of Spreadsheet elements (in
particular, the ability to specify dynamic offsets), time histories, scenarios, and/or multiple realizations can all be output.

**Read more:** Spreadsheet Elements (page 910).

In some cases, you may want to export the final values for selected outputs (e.g., all Stochastics) for every realization to an external file that you can subsequently read into a statistical analysis software package to facilitate post-processing (e.g., detailed uncertainty and/or sensitivity analysis).

You can do this by manually using a key combination when viewing a Raw Multi-Variate Data Table:

![Multivariate Data Table](image)

**Read more:** Viewing a Raw Multi-Variate Data Table (page 684).

**Note:** The default view for a Multi-Variate result is a 2D Scatter Plot. If you are viewing a 2D Scatter Plot (or a different type of Multi-variate display), you can view a Multi-Variate Data Table by pressing the Table button at the top of the display.

Recall that the specific outputs that are displayed in the table are manually selected when you create a Multi-Variate result (with one option being to display all Stochastic elements).

**Read more:** Selecting Outputs for a Multi-Variate Result Display (page 672).

You can export the data displayed in the table by pressing Ctrl+E while viewing it. This will produce a tab-delimited text file (you will be prompted for a filename). The first line is a header line (Realization, followed by the result labels). Units are shown in parentheses after each result label. Each subsequent line consists of the realization number followed by the values. Data is exported using the current precisions setting (as displayed in the table).

**Note:** You can control the number of significant figures displayed in tables from the Results tab of the Options dialog (accessed via Model |Options… from the main menu).

**Read more:** Controlling Significant Figures and Scientific Notation in Result Displays (page 568).

Ctrl+Shift+E adds the full path to the result labels in the header row.
GoldSim exports exactly what is displayed in the table, so that if you choose to use result classification and screening when displaying the table, this is also reflected in the exported file. Similarly, if you define Capture Points, the selected Capture Points will be reflected in the exported file.

**Read more:** Using Result Classification and Screening in Multi-Variate Results (page 686); Viewing Results at Capture Points (page 569).

### Copying and Exporting Result Displays

Once you have created some result displays (in the form of charts or tables), you may want to copy and/or export the results. This is described below.

GoldSim allows you to copy a chart or table to the clipboard. You can then subsequently paste it into another application (e.g., a word processor). Note that charts are pasted as enhanced metafiles, so that the picture can be edited after it is pasted into another application (if you have software that allows you to edit an enhanced metafile).

To copy a chart display window, right-click in the chart, and select **Copy** from the context menu. Alternatively, you can press **Ctrl+C** while viewing the chart.

**Read more:** Using Context Menus in Charts (page 566).

Table data can be copied using **Ctrl+C**. Note, however, that before copying a table, you must select the items of the table that you wish to copy. If your cursor is simply placed in a cell of the table, only that cell will be copied to the clipboard. You can select multiple cells in the table by placing your cursor in one cell and dragging to another location in the table. Selected items will be indicated in black. Double-clicking on the item in the upper left-hand corner of the table selects the entire table.

**Read more:** Selecting Items and Copying Values in Result Tables (page 567).

Once you have copied multiple items from a table, you can readily paste them into a spreadsheet. The table retains its rows and columns in the clipboard, and therefore will be correctly pasted into separate rows and columns in the spreadsheet.

**Note:** If you press the Copy button (or **Ctrl+C**) from the Summary View of a Distribution result, the preview pane chart is copied to the clipboard.

### Exporting a Chart

You can export a chart to a file of a specified format. You can then subsequently open and edit the file in a separate application (e.g., a graphics package).

To export a chart, right-click in the chart, and select **Export...** from the context menu.

**Read more:** Using Context Menus in Charts (page 566).

You will be prompted for the graphics format. The following choices are available:

- Windows Bitmap (BMP)
- Enhanced Windows Metafile (EMF)
- JPEG Image (JPEG)
• Portable Network Graphics (PNG)
• Graphics Interchange Format (GIF)
Chapter 9: Documenting and Presenting Your Model

A mathematical theory is not to be considered complete until you have made it so clear that you can explain it to the first man whom you meet on the street.

David Hilbert

Chapter Overview

A model which cannot be easily explained is a model that will not be used or believed. As a result, GoldSim was specifically designed to allow you to effectively document, explain and present your model. You can add graphics, explanatory text, notes and hyperlinks to your model, and organize it in a hierarchical manner such that it can be presented at an appropriate level of detail to multiple target audiences. This chapter describes how to use these important features.

In this Chapter

The following topics are discussed in this chapter:

- Step One: Organize Your Model!
- Adding Graphics and Text
- Modifying the Appearance of Elements
- Creating, Editing and Viewing Notes
- Adding Hyperlinks to the Graphics Pane
- Manipulating Graphical Objects
- Creating Printed Documentation
- Creating a GoldSim Player File
Step One: Organize Your Model!

The most important way to ensure that your model is easy to explain and present is to spend some time to organize your model.

The sections below provide some guidance and ideas for ensuring that your model is well organized.

When you begin to design your model, you should always have in mind the audiences to whom you will be presenting or explaining the model. Most models will have at least two audiences:

- a high-level audience (managers and decision-makers who are primarily interested in the "big picture" or "bottom line"); and
- a low-level audience (analysts who are also interested in the technical details and assumptions of the model).

Often, there will be multiple low-level audiences, which are interested in different technical aspects of the model. There may also be a "mid-level" audience, which is interested in more than the "big picture", but does not want to get lost in the details.

For a model to be successful (i.e., useful), you must be able to present and explain your model effectively to all of these audiences.

GoldSim provides the tools for you to create a single model that can be explained and presented to multiple audiences. The primary way in which you accomplish this is by organizing your model into a logical, top-down hierarchy.

Perhaps the most important requirement of a well-documented and easily understood model is that it be well-organized.

Containers provide the mechanism in GoldSim by which you can create well-organized models. In most cases, you do this by placing model elements in a "top-down" containment hierarchy in which the level of detail increases as you "push down" into the hierarchy.

In a well organized model, you can imagine that each Container has a specific "message" and a corresponding audience at which it is targeted.

For example, a model's highest level Container might have the following message: "This is the problem I am trying to solve, and here is the overall structure of the model I have built to solve it". Such a Container would be intended for all audiences.

Another Container's message in the model might be "Here are the detailed assumptions regarding process X". This Container's audience would be the analysts or technical personnel interested in that particular aspect of the model.

You "hide" these details in a Container, because they are of no interest to higher level audiences (e.g., managers). Those who are interested, however, can be shown the contents of the Container.

Note, however, that there is no one “right” way to organize a model. To illustrate this, consider the model below:
This is a simple water management model. The primary components are four ponds (Pond1, Pond2, Pond3 and Evaporation_Pond) and a Pipeline though which water flows and is stored. Conceptually, the model is actually quite simple (as we will see below). What makes the model so complex-looking at first glance is that all of the inputs for each of the five primary components are all in the same Container with those components. It makes the graphics pane very difficult to understand.

There are two different (but equally effective) ways we could use Containers to organize this model in a much more effective manner.

In the first, we create a separate Container for each of the primary components. The main Container would then look something like this:

The inside of one of the component Containers (e.g., for Pond_2) would look like this:
Clearly, this model is much easier to understand than the first one!

Although this approach can work quite well for some models, for others, it may not. In particular, if there are multiple logical connections between the various primary components (e.g., Reservoirs) via intermediate auxiliary calculations, the actual movement of material (e.g., the influences representing the flow of water) can be masked by a “spider web” of influences between the Containers.

Fortunately, there is also a very different way to organize such a model. Instead of creating a separate Container for each of the primary components (and its auxiliary calculations and inputs), we could instead just create two Containers. The primary components would be in one Container and all of the auxiliary calculations and inputs would be in another. The Container for the primary components would then look something like this:

The Container with all of the auxiliary calculations and inputs would look something like this:
Clearly, this model is also much easier to understand than the original one. It also has the advantage of highlighting the “flows” of material or information (in this example, the flow of water) between the primary components.

The point here is that Containers can be used to organize your models in different ways and you should think carefully about how you organize your models. Different parts of the model may be organized in different ways, and there is never one “best” way to organize a model. Your goal, however, should be to create a model structure that makes your model easy to explain and present to multiple audiences.

Note: You can always add Containers and move elements between them after your model has been built. In fact, it is quite common to reorganize a model multiple times as it is being built.

Other Suggestions for Organizing Your Model

Read more: Understanding Containers (page 95); Moving Elements Between Containers (page 102).

In addition to structuring your model in a top-down manner, the following suggestions can also help you to create well-organized and easy to understand models:

- Name the elements of your model carefully. If possible, avoid using abbreviations that will not be easily recognized by others.
Adding Graphics and Text

- Use the Description field for all of your elements. Because the Description is displayed in tool-tips, it can be an effective and highly visible documentation tool.

- Don’t hide the details of your model within long input expressions. It is better to add a few additional elements such that the relationships in the model can be shown graphically and are transparent.

- Filter influences in your model if they detract from the presentation. It is often particularly useful to filter influences at the highest model level (to make the diagrams easier to view).

  Read more: Filtering Influences (page 430).

- Modify the color of influences to highlight key relationships.

  Read more: Editing the Appearance of Influences (page 426).

- In some cases, it may be worthwhile to place similar items in a single Container (even if they are referenced throughout the model). For example, you may want to place all constant inputs in one Container named "Constants" or "Key_Assumptions". If it is important that some of these elements also appear in the Container in which they are referenced (in order to add clarity), you could place a clone of the element in that Container.

  Read more: Cloning Elements (page 952).

- It is often useful to create a "Results" Container in the model Root (or perhaps one containment level down) with Result elements for all of the model's key outputs. This allows you to quickly access and display the model's results.

  Read more: Creating and Using Result Elements (page 570).

- You can use the Hyperlink object to provide navigation aids for your model. The Hyperlink object can be used to provide links “jumps” to specified Containers, Dashboards or elements from any location.

  Read more: Adding Hyperlinks to the Graphics Pane (page 760).

Adding Graphics and Text

Adding graphics (an image or a simple schematic) and explanatory text to the graphics pane is one of the most powerful and straightforward ways to document and explain a model.

The Drawing Tools toolbar in GoldSim provides access to a number of buttons for adding images, graphics and text to your model:
The Drawing Tools toolbar is present by default to the right of the graphics pane (although it can be moved to another location):

The same functions can also be accessed under Graphics|Insert in the main menu.

Adding and editing graphic objects, text and images is discussed in the sections below.

Graphic objects (i.e., lines, polylines, polygons, rectangles, curves, closed curves and ellipses) can be added to the graphics pane by using the Drawing Tools toolbar (originally docked to the right of the graphics pane), or by selecting Graphics|Insert from the main menu.

All of the graphic object buttons (or menu options) operate in the same way: when you select a button or a menu option, the cursor changes its appearance (to a drawing cursor). Clicking in the graphics pane with the cursor allows you to insert the object at that point.
You can delete a graphic object by selecting it and pressing the **Delete** button, or by right-clicking on it and selecting **Delete** from the context menu.

**Adding a Line**

To add a Line to the graphics pane:

1. Select **Line** from the Drawing Toolbar or the main menu under **Graphics|Insert**.
2. Place the cursor at the location in the graphics pane where you want the line to start.
3. Left-click and drag the line to the location where you want it to end.

You can move a line by selecting it and dragging it around the graphics pane. You can move one end of the line by selecting the line and dragging one of the end points (vertices).

You can rotate a line by selecting the line and choosing **Graphics|Rotate** from the main menu.

*Read more:*  [Rotating Objects](#) (page 769);  [Changing the Appearance of Graphic Objects](#) (page 746).

**Adding a Polyline**

A Polyline is a segmented line. To add a Polyline to the graphics pane:

1. Select **Polyline** from the Drawing Toolbar or the main menu under **Graphics|Insert**.
2. Place the cursor at the location in the graphics pane where you want the polyline to start, and click.
3. Move the cursor to the next vertex of the polyline and click.
4. After adding as many vertices as desired, move the cursor to the location where you want the polyline to end and double-click.

You can move a polyline by selecting it and dragging it around the graphics pane. You can move a vertex of the polyline by selecting the line and dragging one of the vertices.

You can rotate a polyline by selecting the object and choosing **Graphics|Rotate** from the main menu.

*Read more:*  [Rotating Objects](#) (page 769);  [Changing the Appearance of Graphic Objects](#) (page 746).

**Adding a Polygon**

To add a Polygon to the graphics pane:

1. Select **Polygon** from the Drawing Toolbar or the main menu under **Graphics|Insert**.
2. Place the cursor at the location in the graphics pane where you want one of the corners of the polygon to be and click.
3. Move the cursor to the next vertex of the polygon and click.
4. After adding as many vertices as desired, move the cursor to the location where you want the last vertex of the polygon to be and double-click.

You can move a polygon by selecting it and dragging it around the graphics pane. You can "stretch" a polygon by selecting the object and dragging one of the "handles" (there are eight handles - one on each corner and one on each side). Pressing the **Shift** key while resizing the object keeps the same aspect ratio; pressing the **Ctrl** key while resizing keeps the center of the object at the same location.
You can also move individual vertices by right-clicking on the object and selecting **Edit Vertices** from the menu. This will highlight individual vertices that can then be dragged and repositioned.

You can rotate a polygon by selecting the object and choosing **Graphics|Rotate** from the main menu.

**Read more:** Rotating Objects (page 769); Changing the Appearance of Graphic Objects (page 746).

### Adding a Rectangle

To add a Rectangle to the graphics pane:

1. Select **Rectangle** from the Drawing Toolbar or the main menu under **Graphics|Insert**.
2. Place the cursor at the location in the graphics pane where you want one of the corners of the rectangle to be.
3. Left-click and drag the line to the location of the opposite corner of the rectangle.

You can move a rectangle by selecting it and dragging it around the graphics pane.

You can "stretch" a rectangle by selecting the object and dragging one of the "handles" (there are eight handles - one on each corner and one on each side). Pressing the **Shift** key while resizing the object keeps the same aspect ratio; pressing the **Ctrl** key while resizing keeps the center of the object at the same location.

You can rotate a rectangle by selecting the object and choosing **Graphics|Rotate** from the main menu.

**Read more:** Rotating Objects (page 769); Changing the Appearance of Graphic Objects (page 746).

### Adding a Curve or a Closed Curve

To add a Curve or a Closed Curve to the graphics pane:

1. Select **Curve** or **Closed Curve** from the Drawing Toolbar or the main menu under **Graphics|Insert**.
2. Place the cursor at the location in the graphics pane where you want the curve to start, and click.
3. Move the cursor to the next "handle" of the curve and click.
4. After adding as many handles as desired, move the cursor to the location where you want the curve to end and double-click.

If you are drawing a closed curve, GoldSim will automatically draw a line from the end point to the starting point of the curve (in order to close the curve).

You can move a curve by selecting it and dragging it around the graphics pane. You can stretch the curve by selecting one of the "handles" and dragging it. For a closed curve, you can also move individual vertices by right-clicking on the object and selecting **Edit Vertices** from the menu. This will highlight individual vertices that can then be dragged and repositioned.

You can rotate a curve by selecting the object and choosing **Graphics|Rotate** from the main menu.

**Read more:** Rotating Objects (page 769); Changing the Appearance of Graphic Objects (page 746).

### Adding an Ellipse

To add an Ellipse to the graphics pane:
1. Select **Ellipse** from the Drawing Toolbar or the main menu under **Graphics|Insert**.

2. Place the cursor at the location in the graphics pane where you want one of the corners of a rectangle surrounding the ellipse to be.

3. Left-click and drag the line to the location of the opposite corner of the rectangle.

You can move an ellipse by selecting it and dragging it around the graphics pane. You can "stretch" an ellipse by selecting the object and dragging one of the "handles" (there are eight handles - one on each corner and one on each side). Pressing the **Shift** key while resizing the object keeps the same aspect ratio; pressing the **Ctrl** key while resizing keeps the center of the object at the same location.

You can rotate an ellipse by selecting the object and choosing **Graphics|Rotate** from the main menu.

**Read more:** [Rotating Objects](#) (page 769); [Changing the Appearance of Graphic Objects](#) (page 746).

To edit the properties of a graphic object, double-click on it or right-click on it (to access its context menu), and select **Appearance**…. A dialog like this will be displayed:

![Appearance dialog](image)

The **Fill** tab is not present when editing lines, polylines and curves.

The **Line** tab allows you to edit the appearance of the line itself (for lines, polylines and curves) or the outline for the object (for polygons, rectangles, closed curves, and ellipses). You can select the **Color**, the **Style** and the **Width** (and for Lines and Polylines, the **Line Head**). You can also make the line Transparent (which would only make sense for polygons, rectangles, closed curves, and ellipses).

**Note:** The default Line Head for new Lines and Polylines can be defined in the Graphic tab of the Options dialog (accessed via **Model|Options** from the main menu).

The **Fill** tab (which is not available for lines, polylines and curves) allows you to change the appearance of the fill for the object.
The fill consists of a background and a foreground. The **Foreground Color** is superimposed on top of a **Background Color** according to a selected **Pattern**. By default, the background is transparent.

The **Locking** tab determines what you can do with the object:

If the **Allow Move** and **Allow Rotation** boxes are checked (the default), you can move and rotate the object.

If **Allow Scaling** is checked (the default), you can resize the object. For objects that can be scaled, if **Lock Aspect Ratio** is checked, the object keeps its aspect ratio when it is resized.

If **Lock Appearance** is checked, all of these options are disabled (you cannot move, rotate or scale the object).

Text can be added to the graphics pane by using the Drawing Tools toolbar (originally docked to the right of the graphics pane), or by selecting **Graphics|Insert|Text** from the main menu.

When you select the **Text** button or menu option, the cursor changes to the text cursor. Clicking in the graphics pane with the cursor allows you to insert the text object at that point.

**Note:** GoldSim provides two ways to add text to the graphics pane: a Text object and a Text Box. Text objects are movable objects that can utilize a single font. Text Boxes are scrollable boxes of text that can simultaneously utilize multiple fonts and can be “pinned” to a specific location in the graphics pane.
Adding Graphics and Text

Read more: Adding and Editing Text Boxes (page 751).

To add a Text object to the graphics pane:

1. Select Text from the Drawing Toolbar or the main menu under Graphics|Insert.
2. Place the cursor at the location in the graphics pane where you want the text to be inserted and click.

"<Add your text here.>" is inserted into the graphics pane.

The default font which is used can be selected from the Graphic tab of the Options dialog (accessed via Model|Options in the main menu).

You can move a text object by selecting it and dragging it around the graphics pane. You can rotate a text object by selecting it and choosing Graphics|Rotate from the main menu.

Note: You can resize a text object by selecting it and dragging the "handles". However, when you should always ensure that the text object is sized to allow for some space around (above, below, left, right) the text itself. Otherwise, the text could potentially be cut off at certain resolutions and/or text scaling.

You can delete a text object in the graphics pane by selecting it and pressing the Delete button, or by right-clicking on it and selecting Delete from the context menu.

Read more: Rotating Objects (page 769).

Editing Text Objects

Double-clicking on a text object in the graphics pane selects the text (it will be highlighted in blue). If you start typing while the entire text is highlighted in blue, the text will be overwritten. Once the text is selected, clicking anywhere within the text places the cursor at that location.

You can also use the Home, End, Backspace and arrow keys to move the cursor in the selected text.

A single click on a text object selects the object, but does not select the text itself. Eight "handles" will be visible, defining the limits of the textbox (one at each corner, and one on each side).

By default, word-wrap is on, so the text will wrap around given a specified width. As you type, the textbox will automatically expand downward.

Changing the Appearance of Text Objects

To edit the properties of a text object (e.g., font, fill), right-click on it (to access its context menu), and select Appearance…. The following dialog will be displayed:
The **Font** tab controls the font of the text, and the options are self-explanatory.

The **Text** tab allows you to edit the text and adjust its alignment:

You can edit the **Text** directly in this tab. If **Multiple Lines** is checked (the default), the text object can continue onto multiple lines (either by pressing Enter while editing it or by turning on Word-wrap). The **Word Wrap** checkbox activates word wrap.

---

**Warning:** Word Wrap is ignored if **Multiple Lines** is turned off (cleared). That is, your text will only wrap if you allow for multiple lines.

You can center the text horizontally in the textbox by choosing a **Horizontal Alignment** option. You can center the text vertically in the textbox by choosing a **Vertical Alignment** option.

The **Line** tab allows you to edit the appearance of the outline for the textbox:
By default, there is no outline (it is **Transparent**).

You can change the outline's **Color**, **Style** and **Width**.

The **Fill** tab allows you to change the appearance of the fill for the textbox.

The fill consists of a background and a foreground. The **Foreground Color** is superimposed on top of a **Background Color** according to a selected **Pattern**.

The **Locking** tab determines what you can do with the object:

If **Allow Move** and **Allow Rotation** boxes are checked (the default), you can move and rotate the text. If **Allow Scaling** is checked (the default), you can resize the textbox.
If Lock Aspect Ratio is checked, the textbox keeps its aspect ratio when it is resized. If Lock Appearance is checked, all of these options are disabled (you cannot move, rotate or scale the text).

Text boxes can be added to the graphics pane by using the Drawing Tools toolbar (originally docked to the right of the graphics pane), or by selecting Graphics|Insert|Text Box from the main menu.

When you select the Text Box button or menu option, the cursor changes to the text box cursor. Clicking in the graphics pane with the cursor allows you to insert the text object at that point.

Note: GoldSim provides two ways to add text to the graphics pane: a text object and a text box. Text objects are movable objects that can utilize a single font. Text boxes are scrollable boxes of text that can simultaneously utilize multiple fonts and can be “pinned” to a specific location in the graphics pane.

Read more: Adding and Editing Text (page 747).

To add a text box to the graphics pane:

1. Select Text Box from the Drawing Toolbar or the main menu under Graphics|Insert.
2. Place the cursor at the location in the graphics pane where you want the text box to be inserted, left-click, drag the cursor to specify the size of the box, and release the mouse button.

At this point the text box will be inserted into the graphics pane and the dialog for editing the text box will be displayed:

After adding text, pressing the Close button closes the dialog.

You can double-click on the dialog to edit it again. You can move a text box by selecting it and dragging it around the graphics pane. You can resize the text box by grabbing and dragging one of the handles of the box. You can delete a text box in the graphics pane by selecting it and pressing the Delete button, or by right-clicking on it and selecting Delete from the context menu.
Adding Graphics and Text

Note: All of these operations (editing, moving, resizing, deleting) can only occur if the size and position are not locked (i.e., the Lock size & position checkbox is cleared).

Controlling the Behavior of Text Boxes

You can control the format of the text in a text box by using the buttons and controls at the top of the dialog. These buttons and controls are self-explanatory (and all have tool-tips), and allow you to control the font, size, color, and justification of the text. Note that the text can have multiple fonts, formats and colors. You can also control these options by right-clicking on the text and selecting options from a context menu.

Note: The default font used within the text box can be selected from the Graphic tab of the Options dialog (accessed via Model|Options in the main menu).

Two checkboxes at the bottom of the dialog control its behavior:

If the Border checkbox is checked, a border is placed around the textbox.

The Lock size & position checkbox switches the text box between display mode and edit mode. If this box is checked, a vertical slider appears in the text box (if necessary), and you cannot move, resize, edit or delete the text box:

If the Lock size & position checkbox is cleared, the text box can be edited by double-clicking on it, can be moved by selecting and dragging, can be resized by dragging the handles, and can be deleted by selecting and pressing the Delete key. A text box which is not locked appears in the graphics pane like this:

You can unlock a text box in the graphics pane by right-clicking on it and clearing the Lock size & position menu item. Likewise, you can lock a text box
Adding Graphics and Text

Inserting Hyperlinks into Text Boxes

You can insert hyperlinks from a text box to other documents:

- To insert a link to a URL which starts with "www", you need only type in the address (e.g., www.goldsim.com).
- To insert a link to a URL which does not start with "www", you must specify the transfer protocol (e.g., http://website.com/).
- To insert a link to an ftp site, you need only type in the address (e.g., ftp.asite.com).
- To insert a link that launches an email, you must type “mailto:” before the email address (e.g., mailto:name@company.com).
- To insert a link to some other document (e.g., a Word document or a PowerPoint presentation), you must type "file://" before the document name and one of the following:
  - The full path (e.g., file://C:\Projects\Docs\report.doc).
  - A relative path (e.g., file://..\Docs\report.doc), in which case GoldSim searches relative to the current directory (the directory from which the model file was loaded). If the model file was in C:\Projects\Models, GoldSim would look for the document in C:\Projects\Docs.
  - No path (e.g., file://report.doc), in which case GoldSim looks only in the current directory (the directory from which the model file was loaded).

When you are entering a hyperlink, GoldSim recognizes it as a link as soon as it sees a space (it will become underlined and the text will become blue). As a result, when you are entering the text, there cannot be any spaces in the link. If the filename includes a space, you can represent this by using the characters %20 to represent the space. For example, if the filename you wanted to link to was “this file.pdf”, the link in the Note pane would need to look like this:

```
File://this%20file.pdf
```

When you click on a hyperlink in a text box, GoldSim will immediately open the application associated with the link (e.g., Internet Explorer, Word).

Adding Images

In addition to adding text and graphics, you can also add images (as bitmaps and other graphic formats) to the graphics pane.
To add an image to the graphics pane:

1. Select Image from the Drawing Toolbar or the main menu under Graphics|Insert. A dialog is displayed for selecting the image file to be inserted.
2. Select a file and file type to insert and press OK.
3. Place the cursor at the location in the graphics pane where you want the center of the image to be inserted and click.

The image will be inserted into the graphics pane. You can move an image by selecting it and dragging it around the graphics pane.

You can "stretch" (resize) an image by selecting it and dragging one of the "handles" (there are eight handles - one on each corner and one on each side). Pressing the Shift key while resizing the image keeps the same aspect ratio; pressing the Ctrl key while resizing keeps the center of the object at the same location.

GoldSim provides some capabilities for you to edit the properties and appearance of an image that you have inserted into the graphics pane.

To edit the properties of an image in the graphics pane, right-click on it (to access its context menu), and select Appearance…. The following dialog will be displayed:

If the Allow Move box is checked (the default), you can move the image.

If Allow Scaling is checked (the default), you can resize the image. If Lock Aspect Ratio is checked, the image keeps its aspect ratio when it is resized (scaled). If Lock Appearance is checked, both of these options are disabled (you cannot move or scale the image).

**Note:** If your image is an enhanced metafile, the dialog also contains a Line and a Fill tab. The Line tab allows you control the appearance of the outline of the image. The Fill tab allows you to control the appearance of the background of the image.
Modifying the Appearance of Elements

All of the GoldSim elements have default symbols (graphical images by which the element is represented in the graphics pane).

Read more: GoldSim Element Types (page 68).

One of the most powerful ways to document and visually augment your model is to replace the default symbols with customized symbols. This is particularly useful for Containers. For example, you may want to replace the image of a particular Container with a schematic or diagram of the subsystem that it represents.

Read more: Editing the Appearance of Elements (page 434).

Creating, Editing and Viewing Notes

In order to internally document your model, GoldSim allows you to attach a Note to each element:

An element with an attached note has its ID underlined (and by default the text is blue).

The topics below describe how to create, edit and display notes.

To display, create and edit notes, you must activate the Note pane. By default, the Note Pane is turned off (hidden). The Note pane can be toggled on and off in order to edit and view notes for your elements.

You do this by pressing the Note Pane button in the Standard toolbar, by selecting View|Note from the main menu, by pressing Ctrl-Shift-N, or selecting Note View from the toolbar context menu (accessed by right-clicking anywhere outside of the graphics pane and browsers).

You can also open the Note Pane for an element by clicking on the element ID for an element with an attached Note (indicated by an underline).

By default, the Note Pane is initially docked on bottom of the screen:
The Note Pane can be in four different states (Floating, Docking, Auto Hide and Hide), which can be selected from the drop-list at the top of the Note Pane:

You can make the Note Pane become a floating window by selecting **Floating** from the drop-list (or double-clicking on the Note Pane title bar):
You can dock a floating Note Pane (to any side of the screen) by starting to drag it. When you do so, multiple docking points appear on the screen:

Dragging to the docking point docks the Note Pane at that location.
You can hide the Note Pane completely by selecting Hide from the drop-list.
You can also hide the browser by pressing Ctrl-Shift-N or by pressing the Note Pane button in the Standard toolbar:

The final option is to Auto Hide the Note Pane. If you select this option (which is not available if the Note is floating), the Note Pane is hidden until you move
Creating and Displaying Notes

the cursor over the section marked “Note View” on the side of the screen where
the Note Pane is docked:

When you do so, the Note Pane opens (and stays open until you click the cursor
moves outside of the Note Pane).

Whenever you select an element, the Note for that particular element is
displayed in the Note Pane. The element whose Note is being displayed is
indicated in the Note Pane. If nothing is selected in the graphics pane, the Note
for the entire Container is displayed.

To add a note to an element, do the following:

1. Select the element (by clicking on it).
2. Open the Note Pane by pressing the Note Pane button in the Standard
toolbar or by selecting View|Note from the main menu.
3. Click in the Note Pane and start typing.

The default font for a new note is specified in the Graphic tab of the Options
dialog (accessed via Model|Options in the main menu).

A Format Toolbar is available that provides tools for formatting your Note. By
default, the Format Toolbar is turned off. You can show and/or hide the Format
Toolbar by right-clicking in the Note Pane and selecting Show Format Bar
from the menu.

Note: The basic settings for the Note pane (its location, size and whether or not
the Format Toolbar is visible) is saved to the Windows Registry, and are
therefore applied to all GoldSim files on your machine. Formatting for specific
Notes (e.g., font, color), however, is saved only with the file.

You can only enter a Note if an element is selected (or nothing is selected). If
nothing is selected, the Note is attached to the Container being viewed. If a
graphical object is selected, you will not be able to type in the Note pane (since
Notes can only be associated with elements).

In addition to typing in the Note pane, you can paste text from another Note or
another application (such as a word processor).

Note: Notes in GoldSim support multibyte characters sets (for supporting
character sets, such as Japanese and Chinese, which cannot be represented in a
single byte).

Once a Note has been created for an element, the element’s ID will be
underlined in the Graphics Pane and a single click on the element ID in the
Graphics Pane will open the Note pane and display the Note for that element:

By default, the ID of an element with a Note associated with it is underlined and
shown in blue (so that it looks like a hyperlink). You can modify this so that the
ID is shown in the defined text color (black by default) via an option on the
Creating, Editing and Viewing Notes

**General** tab of the Options dialog (accessed by selecting **Model|Options** from the main GoldSim menu).

To delete a note attached to an element, simply delete all of the text in the note.

The Note Pane acts like a simple text editor. That is, you can specify alignment, fonts, font format, text color, indents, and create simple numbered and bulleted lists.

You can access formatting tools for your note in two ways:

- You can show and/or hide the Format Toolbar by right-clicking in the Note Pane and selecting **Show Format Bar** from the context menu:

  ![Format Toolbar](image1.png)

- You can view a context menu with all formatting options by right-clicking anywhere within the Note pane:

  ![Context Menu](image2.png)

The various formatting options in the toolbar and menu are self-explanatory.

Note also that you can use this menu to Print the note.

**Inserting Hyperlinks into Notes**

One of the most powerful features of Notes is that you can insert hyperlinks to other documents:

- To insert a link to a URL which starts with "www", you need only type in the address (e.g., www.goldsim.com).
- To insert a link to a URL which does not start with "www", you must specify the transfer protocol (e.g., http://website.com/).
- To insert a link to an ftp site, you need only type in the address (e.g., ftp.asite.com).
- To insert a link that launches an email, you must type “mailto:” before the email address (e.g., mailto:name@company.com).
Adding Hyperlinks to the Graphics Pane

To insert a link to some other document (e.g., a Word document or a PowerPoint presentation), you must type "file://" before the document name and one of the following:

○ The full path (e.g., file://C:\Projects\Docs\report.doc).

○ A relative path (e.g., file://..\Docs\report.doc), in which case GoldSim searches relative to the current directory (the directory from which the model file was loaded). If the model file was in C:\Projects\Models, GoldSim would look for the document in C:\Projects\Docs.

○ No path (e.g., file://report.doc), in which case GoldSim looks only in the current directory (the directory from which the model file was loaded).

When you are entering a hyperlink, GoldSim recognizes it as a link as soon as it sees a space (it will become underlined and the text will become blue). As a result, when you are entering the text, *there cannot be any spaces in the link*. If the filename includes a space, you can represent this by using the characters %20 to represent the space. For example, if the filename you wanted to link to was “this file.pdf”, the link in the Note pane would need to look like this:

![Link Example](image)

When you click on a hyperlink in a note, GoldSim will immediately open the application associated with the link (e.g., Internet Explorer, Word).

Adding Hyperlinks to the Graphics Pane

You can also add Hyperlink objects to the graphics pane. When you double-click on a hyperlink in the graphics pane, it can open another application (e.g., a website, a PowerPoint presentation, an email program, a Word document, a graphics image), or jump to a specific location in your model (which can be useful for navigation of complex models).

To insert a hyperlink, press the **Hyperlink** button on the Drawing Tools toolbar:

![Hyperlink Button](image)

Alternatively, you can select **Graphics|Insert|Hyperlink…** from the main menu.

The following dialog will be displayed:
The **Label** appears below the Hyperlink object in the graphics pane. Note that unlike element IDs, the label can have spaces (but it cannot be completely empty).

The **Type** field contains the type of hyperlink you wish to make in the **Address** field. GoldSim provides the following options:

- **Web or local document**: Opens a website or a document (e.g., a Word or PowerPoint document);
- **Email to...**: Opens your default email program and inserts the specified email address in the “To” field;
- **FTP**: Opens an FTP site;
- **Execute Command**: Opens a specific program;
- **Go to Element**: Jumps to a specific element and selects it. This is useful for navigating a model.
- **Open Container**: Jumps to a specific Container. This is useful for navigating a model.
- **Open Dashboard**: Jumps to a specific Dashboard. This is useful for navigating a model.

In the latter three cases, after selecting the option, you must specify the element, Container or Dashboard that you wish to link to:

You can also enter a **Tooltip**. This is displayed when you place your cursor over the Hyperlink object in the graphics pane.

If the **Editable** check box is checked (the default), you will be able to move the Hyperlink object around the graphics pane (and resize it). If this checkbox is cleared, the object will be “pinned” to the location where you placed it. You can make the object editable by right-clicking on it in the graphics pane and checking the **Editable** menu item. Likewise, you can lock (make uneditable) an editable hyperlink by right-clicking on it in the graphics pane and clearing the **Editable** menu item.
After entering the Hyperlink’s properties in the dialog and pressing **OK**, you will be presented with a “placement cursor”. When you click with the cursor, the Hyperlink object is inserted at that location. The Hyperlink object looks like this:

![Reference](Image)

You can access the dialog for editing the Hyperlink’s properties by selecting **Properties…** from the context menu for the object.

When you double-click on a Hyperlink object, GoldSim will immediately open the application associated with the link (e.g., Internet Explorer, Word).

You can move a Hyperlink object by selecting it and dragging it around the graphics pane. You can resize a hyperlink by selecting the object and dragging one of the “handles” (there are eight handles - one on each corner and one on each side). Note, however, that these operations are only available if you have checked the Hyperlink as being Editable.

The details regarding how the various Hyperlink types are specified are provided below:

**Web or local document**: To insert a link to a URL which starts with www, you need only type in the address (e.g., www.goldsim.com). To insert a link to a URL which does not start with www, you must specify the transfer protocol (e.g., http://website.com).

To insert a link to some other document (e.g., a Word document or a PowerPoint presentation), you can provide the full path (e.g., C:\Docs\Projects\report.doc), or if the document is in the same directory as the model file, you can provide just the file name. That is, if you only provide a file name, GoldSim will look for the document in the active directory (the directory containing the current model file). You can also use the Browse… button to find a particular file to which you wish to create a hyperlink.

One common type of document that you may wish to open is a Microsoft Excel file (*.xls, *.xlsx or *.xlsm). When opening a spreadsheet file, in some cases you may want to immediately jump to a specified location (sheet and cell range) when opening the file. GoldSim provides specialized syntax to support this. The syntax is as follows:

Filename#sheetname!cellrange (e.g., Book1.xlsx#Sheet1!a1:c3)

The following points should be noted:

- The entire text can be wrapped in quotes. This should be done if the filename or sheet name contain any spaces.
- For an Excel file, GoldSim will recognize extension .xls, .xlsx or .xlsm.
- The filename can be specified as a relative or absolute path. Absolute filenames must start with the drive letter or be a UNC path (starting with “\”). Relative paths can contain “.” and “..” as well as folder names (e.g., “..\DataFolder\file.xlsx”). Relative filenames with no path are assumed to refer to the folder in which the model file is located.
Adding Hyperlinks to the Graphics Pane

- # is the delimiter between the filename and the sheet name. It is optional (you do not need to provide a sheet name). The sheet name is not case-sensitive.

- ! is the delimiter between the sheet name and the cell range. It is optional (you do not need to provide a cell range). The cell range can refer to a single cell (e.g., A2) or a range (e.g., A1:A3).

**Email to...** When you click on an email link, it opens your default email program with the specified address in the “To” field. You should enter an email address (e.g., support@goldsim.com).

**FTP:** To insert a link to an ftp site, simply type in the address (e.g., ftp.asite.com).

**Execute Command:** This allows you to run a specific application. This can be useful, for example, if you wish to open a document with an application which is not the default application with which the file extension is associated:

```
c:\graphics\paint.exe c:\images\bitmap.doc
```

Note that when using this type of Hyperlink, if the filename has spaces in it, it must be enclosed in quotation marks:

```
c:\graphics\paint.exe "c:\images\my bitmap.doc"
```

**Go to Element.** This causes GoldSim to immediately jump to (but not open) the specified element. You can also specify a full path when using the element command. When you do so, the path name must start with a backslash:

```
\Container1\Container2\XYZ
```

**Open Container.** This causes GoldSim to immediately display the contents of the specified Container. GoldSim searches upward through the containment hierarchy from the current location and jumps to the first Container with the specified name. If you need to be more explicit (i.e., you have multiple Containers with the same name), you can also specify a full path when using the Container command. When you do so, the path name must start with a backslash:

```
\Container1\Container2\Container3
```

**Open Dashboard.** This causes GoldSim to immediately jump to the specified Dashboard. You can also specify a full path when using the Dashboard command. When you do so, the path name must start with a backslash:

```
\Container1\Container2\InputScreen
```

**Note:** Dashboards are discussed in detail in the GoldSim Dashboard Authoring Module User’s Guide.

**Changing the Appearance of a Hyperlink Object**

You can edit the appearance of a hyperlink by right-clicking on the object and selecting **Appearance...** from the context menu. The following dialog will be displayed:
Adding Hyperlinks to the Graphics Pane

Note: You can only change the appearance of a Hyperlink object if it is specified as being Editable. Editing can be enabled and disabled by right-clicking on the object and setting or clearing the Editable menu item.

The dialog is tabbed. The various tabs are discussed in the sections below.

Changing the Hyperlink Object’s Symbol

The Image tab of the Hyperlink Appearance dialog allows you to change the image used for the Hyperlink object in the graphics pane.

If you select the User-defined symbol radio button within this tab, you will be prompted for the name of a file to be used for the symbol. The file must be an enhanced Windows metafile (.emf).

Note: GoldSim utilizes enhanced metafiles for symbols because they are vector graphics (and therefore scale without losing image quality). Most advanced graphics programs can create enhanced metafiles (or convert other formats to an enhanced metafile). Note, however, that unless you create the original image as an enhanced metafile, it will not be a true vector graphic and will lose image quality when scaling.

You can use GoldSim’s graphic capabilities to convert other graphics formats to an enhanced metafile as follows:

1. Insert a graphic image (e.g., a bitmap) into the graphics pane;
2. Select the image and click **Graphics | Export…** from the main menu (or press **Ctrl+E**);

3. Specify that you wish to save the file as an enhanced metafile.

*Read more:* [Adding Images](page 753).

Once the symbol is defined using a user-defined symbol, you can change the symbol by pressing **Select new Image…**, or you can switch back to the default image provided by GoldSim by selecting the **Default symbol** button.

You can access the dialog for selecting a new image directly (without using the Appearance dialog) by selecting **Change Symbol…** from the context menu for the element.

Two additional buttons are also on the **Image** tab:

- **Reset Size**: This resets the size of a scaled image back to its original size.
- **Reset Ratio**: This resets the aspect ratio of a scaled image back to its original ratio (by changing the symbol’s height).

Two tabs within the Hyperlink Appearance dialog control the appearance of the object's label. The **Font** tab controls the label's font.

The options in this dialog are self-explanatory.

The **Labels** tab controls the position of the label relative to the symbol.

You can choose one of nine positions for the label. Note that if one of these nine positions is not sufficient, you can also manually move the label by selecting an **X-Offset** and/or a **Y-Offset** (these can be negative numbers). In addition, if
Allow Label Move is checked in the Locking tab, you can select and manually move the label.

Two tabs within the Hyperlink Appearance dialog control the appearance of the object's background and outline. The Line tab allows you to draw a line (a box) around the symbol and label.

By default, the line is transparent. If you clear this box, you can change the line's color, style and width. Note, however, that the style can only be applied to 0 point lines.

The Fill tab allows you to change the appearance of the fill for the symbol and label.

The fill consists of a background and a foreground. The Foreground Color is superimposed on top of a Background Color according to a selected Pattern. By default, both are transparent.

The Locking tab determines what you can do with the object:
If Allow Move and Allow Rotation boxes are checked (the default), you can move and rotate the text. If Allow Scaling is checked (the default), you can resize the textbox. If the Allow Label Move box is checked you can select and move the object’s label (separately from the object itself). If the box is cleared (the default), the label can only be moved with the object itself.

If Lock Aspect Ratio is checked, the textbox keeps its aspect ratio when it is resized. If Lock Appearance is checked, all of these options are disabled (you cannot move, rotate or scale the object).

**Manipulating Graphical Objects**

GoldSim provides a number of utilities for aligning, ordering, spacing, sizing, grouping, rotating, pasting and moving graphical objects (including element symbols) in the graphics pane.

If you have selected multiple objects and/or elements in the graphics pane, you can choose Graphics|Align from the main menu to access a submenu.

The options in this menu (which are self-explanatory) can be used to align the selected objects.

When you add an object to the graphics pane, it is assigned a particular order. The order determines how objects are "stacked" (i.e., whether an object is in front or behind another object) when they overlap. Higher order objects are always placed in front of lower order objects:

With regard to order, GoldSim groups objects into three classes: 1) elements; 2) influences; and 3) everything else (graphical objects like images and text). Elements always have a higher order than influences and influences always have a higher order than graphical objects. Hence, influences will always appear behind elements, and images and text will always appear behind influences.

Within a given class of objects (i.e., elements, influences or graphical objects), the latest object added to the graphics pane always has the highest order. Hence, if you add three circles, the first one added will have the lowest order and the last one added will have the highest order.
Once an object is placed in the graphics pane, you can manually change its order (within its class of objects) by selecting the object and choosing Order from the context menu to display a submenu.

- **Bring to Front** brings the object all the way to the front (gives it the highest order).
- **Send to Back** sends the object all the way to the back (gives it the lowest order).
- **Bring Forward** increases the object's order by one.
- **Send Backward** decreases the object's order by one.

The order and align options are also available in the Graphical Tools toolbar:

### Spacing and Sizing Objects

If multiple objects are selected, you can also choose to space them evenly (horizontally or vertically), or set their widths, heights or sizes to be identical. These options are provided under **Graphics|Layout** in the main menu:

The layout options are also available from the Graphical Tools toolbar:
Precisely Moving Objects

Although you can move objects around the graphics pane by simply dragging them from one location to another, in some cases you may want to move them in a more precise manner. To facilitate this, GoldSim allows you to move objects vertically and horizontally using the arrow keys.

There are two modes of movement using the arrow keys:

- Using the arrow keys alone moves the object a short distance in a horizontal or vertical direction.
- Holding the Shift key down while using the arrows increases the distance moved by a factor of five.

Note also that when moving objects in the graphics pane, you can force their upper left hand corner to snap to a grid.

Read more: The Graphics Pane Grid and Background (page 423).

Grouping Objects

If you have selected multiple objects in the graphics pane, you can choose Grouping|Group from the context menu of one of the objects or Graphics|Group|Group from the main menu to group the objects. When objects are grouped, they behave as a single object. Hence, when you select one object, all objects in the group are selected. When you edit the graphical property (e.g., font, background color) of one object, the corresponding property of all objects in the group is modified.

Note: Elements cannot be a member of a group. Only graphical objects, text, images and hyperlinks can be members of a group.

To Ungroup the set of objects, select the Group, and choose Grouping|Ungroup from the context menu of one of the objects or Graphics|Group|Ungroup from the main menu.

The Group and Ungroup options are also available in the Graphical Tools toolbar:

Rotating Objects

You can rotate a graphic object or text object in the graphics pane by selecting the object and choosing Graphics|Rotate from the main menu to access a submenu.

- **Free Rotate** activates a Rotate Cursor (when placed over the object). Dragging the cursor within the object rotates it about its center point.
- **Rotate Left** rotates the object 90 degrees counter clockwise.
- **Rotate Right** rotates the object 90 degrees clockwise.
- **Flip Horizontal** flips the object horizontally (around a vertical center axis).
- **Flip Vertical** flips the object vertically (around a horizontal center axis).
The Rotate options are also available in the Graphical Tools toolbar:

![Graphical Tools toolbar](image)

**Note:** If an element is selected, the rotate option will be unavailable (grayed out). That is, you cannot rotate an element.

### Copying, Pasting, Moving and Deleting Graphical Objects

You can copy and paste objects within the graphics pane using Ctrl+C and Ctrl+V, respectively.

You can delete an object by selecting it and pressing the **Delete** key.

You can also use the context menu for the object to **Cut**, **Copy**, **Paste** or **Delete**.

You can paste graphical objects and text from one GoldSim model to another and/or from other applications into GoldSim. For example, you could paste a bitmap from a graphics program directly into the GoldSim graphics pane, or you could paste text from a word processing application or text editor directly into the graphics pane.

When you paste text from another application into the GoldSim graphics pane, you must tell GoldSim how it is to treat any tabs that are present in the pasted text, and the font for the text. This is specified in the **Graphic** tab of the Options dialog (accessed via **Model|Options…** from the main menu).

![Options dialog](image)

You can also move graphical objects from one container to another in the same manner that you can move elements.

**Read more:** [Moving Elements Between Containers](#) (page 102).
Creating Printed Documentation

GoldSim provides an automated "undo" and "redo" capability for actions involving the manipulation of graphics. These are accessible from the main menu (Edit|Undo and Edit|Redo). Alternatively, you can use Ctrl+Z and Ctrl+Y for Undo and Redo, respectively.

The Undo function only operates on actions involving the manipulation of graphic objects, such as adding, moving or modifying graphics and text. It cannot Undo actions that affect the model itself (such as editing inputs to an element).

The "stack" of actions that can be undone is terminated as soon as you carry out an action which cannot be undone. For example, if you add a text object, then add a rectangle, and then edit its graphical properties, you can step backward and undo these actions. If you subsequently added a new element, however, the "stack" of graphical actions would be deleted (and could not be undone using the Undo function).

The Redo function is only available after you Undo an action. That is, its function is to reverse an Undo.

Creating Printed Documentation

GoldSim provides two mechanisms to create printed documentation of your model:

- You can print the graphics pane;
- You can export the graphics pane (or one or more selected objects) to a graphics file; and
- You can create a simple XML file that can be viewed in a text editor, browser or XML editor that echoes out all of the elements in a GoldSim model.

If you scroll far enough horizontally or vertically you will eventually reach the "edge" of the graphics pane. That is, the graphics pane represents a document of fixed size. This means that an element on the screen scales to a specific size on paper, and the entire graphics pane maps to a printed document with specific dimensions.

Read more: Adjusting the Size of the Graphics Pane (page 425).

In order to print the document, you must specify the manner in which the document is to be represented on paper. In particular, you must specify the paper size and orientation, along with the margins. You do this from a dialog accessed File|Page Setup... on the main menu:
Creating Printed Documentation

When you make a change to the Page Setup (and press OK), GoldSim will ask if you want to apply these settings to all Containers, or just the current Container.

You can view the Page Bounds within the graphics pane by selecting Properties… from the context menu for the graphics pane (and then checking Show Printer Page Boundaries in the Graphics tab that is displayed).

You can preview the printed document by selecting File|Print Preview from the main menu. You can print the graphics pane directly from the Print Preview window or by selecting File|Print from the main menu.

You can export the graphics pane and save it to a graphics format.

The simplest way to do this is to use Alt+PrtSc to cut the entire window to the clipboard (as a bitmap).

You can also use GoldSim’s Export utility to save just the graphics pane (or selected items in the graphics pane) to a particular graphics format.

To do so:

1. Select an object (or objects) in the graphics pane, and select Graphics | Export… from the main menu (or press Ctrl+E). A dialog for specifying a file type and a location for the file is displayed.

2. Select a file type and a location, and then press Save.

If no object is selected in the graphics pane when you choose Graphics | Export… from the main menu (or press Ctrl+E), the entire graphics pane will be exported.

You can create a simple XML file that can be viewed in a text editor, browser or XML editor that echoes out all of the elements in a GoldSim model. This file contains the name, type, description and path for each element, and specifically illustrates the model hierarchy. It does not, however, include the definition (e.g., input values, units) for any elements.

You can create this file by selecting File | Save Inventory… from the main menu.

XML stands for eXtensible Markup Language. It was designed as a software and hardware independent tool to store and transport data such that it is both
human- and machine-readable. Like HTML, XML uses tags. However, unlike HTML, XML does not have predefined tags. Rather, the author defines custom tags (as well as the document structure).

The general concept behind XML is to create a tree structure which starts at a root object and branches to child object. Such a structure, of course, is ideal for displaying the hierarchical design of a GoldSim model.

Objects that contain other tags as “contents” (such as Containers in GoldSim) are identified and formed by using opening and closing tags. The closing tag is identical to the opening tab with a forward slash (e.g., <Global>, </Global>).

The tags used by GoldSim are as follows:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;GSM&gt;</td>
<td>Opening and closing tags for entire file.</td>
</tr>
<tr>
<td>&lt;Global&gt;</td>
<td>Demarcates global information regarding file.</td>
</tr>
<tr>
<td></td>
<td>&lt;File&gt;, &lt;Author&gt;, &lt;Created&gt; and &lt;Modified&gt; tags are always inside &lt;Global&gt; opening and closing tags.</td>
</tr>
<tr>
<td>&lt;File&gt;</td>
<td>GoldSim filename and path.</td>
</tr>
<tr>
<td>&lt;Author&gt;</td>
<td>Model Author.</td>
</tr>
<tr>
<td>&lt;Created&gt;</td>
<td>Date and time GoldSim file was created.</td>
</tr>
<tr>
<td>&lt;Modified&gt;</td>
<td>Date and time GoldSim file was last modified.</td>
</tr>
<tr>
<td>&lt;Model&gt;</td>
<td>Demarcates information defining the model itself.</td>
</tr>
<tr>
<td></td>
<td>&lt;Container&gt;, &lt;Elements&gt;, and &lt;Element&gt; tags are always inside &lt;Model&gt; opening and closing tags.</td>
</tr>
<tr>
<td>&lt;Container&gt;</td>
<td>A GoldSim Container. This tag also includes a number of properties (e.g., the Container name).</td>
</tr>
<tr>
<td>&lt;Elements&gt;</td>
<td>Demarcates a list of elements inside a GoldSim Container.</td>
</tr>
<tr>
<td>&lt;Element&gt;</td>
<td>A GoldSim element. This tag also includes a number of properties (e.g., the element name).</td>
</tr>
</tbody>
</table>

The best way to illustrate this is to examine a very simple model file. Consider the following simple model (as illustrated in the browser):

![Browser](image)

As can be seen, in this model, Container1 contains a Data element named A, an Expression element named B, and a Container named Container2. Container2, in turn, contains a Data element named C.

XML files can be viewed in any text editor. The “raw” XML Model Inventory file for the model above as viewed as text looks like this:
The XML Model Inventory file can be viewed in most browsers, and by default will look similar to this (this is viewed using Chrome):

```
<xml version="1.0" encoding="UTF-8" standalone="yes"/>
<xml name="GoldSim"
    dateGenerated="12/11/2003"
    created="5/22/2008 1:02:00 PM" modified="3/25/2003 5:08:19 PM">
    <Global>
        <Model typeID="Model" desc="The base container for the project" path="/">
            <Elements>
                <Container typeID="Container" type="Container" id="Container1" desc="Container1 Description" path="/">
                    <Elements>
                        <Element type="Data" typeID="1" id="1" desc="A Description" path="/Container1/">
                            <Element type="Data" typeID="2" id="2" desc="B Description" path="/Container1/">
                                <Container typeID="Expression" typeID="expression" id="expression" desc="Expression Description" path="/Container1/">
                                    <Elements>
                                    </Elements>
                                </Container>
                            </Element>
                        </Element>
                    </Elements>
                </Container>
            </Elements>
        </Model>
    </Global>
```

Note that when viewing an XML file in a browser, you can expand and collapse branches of the tree. In Chrome, this is done using the arrowheads at each branch (in Internet Explorer, the branches are opened and closed using + and - signs). In the example above, all of the branches are expanded.

As can be seen, the file structure is quite simple. The following should be noted:

- The `<Elements>` tag surrounds all of the elements inside a particular `Container`.
- Every element is described using a tag that has the following format:
  - `<Element type="element type" typeID="element type number" id="element name" desc="element description" path="element path"/>
  - If the element is the `Container`, a `<Container>` tag is used (with the same format as the `<Element>` tag).
  - In addition to the type of element being specified by name, it is also specified by a define `typeID`. The `typeID` is an integer; `typeIDs` for all elements are listed below:
XML files can be formatted by editing the file and specifying a stylesheet (when the XML file is created by Goldsim, no stylesheet is specified). This allows the files to appear much more readable. For example, this is how the XML file for the model above appears if a stylesheet is used that formats the XML file emphasizing the containment hierarchy:
Alternatively, this is how the XML file for the model above appears if a stylesheet is used that formats the XML file by element type:
Creating a GoldSim Player File

The GoldSim Player is a special version of GoldSim that allows you to "play" or "read" an existing GoldSim model without having to license the GoldSim software. In general, the user interface for the GoldSim Player is identical to that of the full GoldSim version, with a number of menu options and controls for editing the model removed or disabled.

The GoldSim Player can be downloaded (for free) from the GoldSim website. It is also automatically installed on your machine when you install GoldSim (and is available via the Windows Start menu).

The GoldSim Player cannot read a normal GoldSim file (a .gsm file). Rather, the GoldSim Player can only read Player files (.gsp). You can save a copy of a GoldSim model as a Player file by selecting `File|Save Player File...` from the main menu.
Creating a GoldSim Player File

When you do so, a wizard will appear to assist you in creating the Player file. The first page of the wizard simply provides some explanatory text. The second page of the wizard prompts you for the Author’s Name, a Model Description, and a Model Title. The Author’s Name, Model Title, and Model Description can be viewed via a menu item in the Player (and the Model Title also appears in the title bar of any Dashboards you create). The third page of the wizard allows you to specify the properties of the Player file:

If the **Allow to browse the model** option is checked, the Player user is allowed to leave any Dashboard(s) and browse the structure of the model. Note that if the model does not have at least one Dashboard and/or the Dashboard includes a Button control that jumps to a Container or an element, the model is automatically browsable and this option will be checked on and grayed out.

If the **Allow the user to run the model** option is checked, the Run Control toolbar within the Player will be active and the user can use this to run (and save) the model. If this option is off, the Run Control toolbar is grayed out and the Player user will be able to view the model, but will not be able to run (or save) it. If the user can run the model, you can also choose whether to **Allow changes to the Time Settings** and **Allow changes to the Monte Carlo settings**. These allow the Simulation Settings of the model to be edited within the Player.

The final field in the third page of the wizard determines what will be displayed when the Player user opens the file. If the Player file has no Dashboards, the only option is “Top-level Container”. However, if the file contains one or more Dashboards, you can specify the **Default Dashboard**; that is, the Dashboard that will be displayed when the file opens in the Player. The drop-list contains all Dashboards in the model. If the model is browsable, it also contains “Top-level Container”.

The final page of the wizard is used to specify the path and filename of the Player file.

Player files can be saved with or without results. If you try to create a Player file while you are in Edit Mode, GoldSim will first check the model for errors. If it finds errors, it will not allow you to create the Player file.

A Player file cannot be read by GoldSim; it can only be read by the GoldSim Player. You can view and navigate a Player file using the GoldSim Player just as you would view and navigate a GoldSim model file in GoldSim. This includes the ability to open most GoldSim dialogs (e.g., in order to examine how an element has been defined).

**Note:** If the model file from which the Player file is created was password-protected, the Player file will also be password-protected, and the user will need to enter a password to open the file.

**Read more:** Password-Protecting a Model File (page 65).

As pointed out above, the user interface for the GoldSim Player is identical to that of the full GoldSim version, with a number of menu options and controls for editing the model removed. Instructions for using the GoldSim Player are described within the program’s online help system, as well as the GoldSim Player User’s Guide.
The GoldSim Dashboard Authoring Module provides tools that allow a GoldSim modeler to design and construct a "dashboard" interface for models.

The interfaces can be designed to look like "dashboards" or "control panels", with buttons, gauges, sliders and display panels, and the author can embed instructions, tool-tips and graphics to provide instructions on the use of the model.

Such an interface allows a model to be easily used by someone without requiring them to be familiar with either the GoldSim modeling environment or the details of the specific model.

When such “dashboarded” models are saved as Player files, Player users can not only view and navigate the model, but they can also modify the model (through the dashboards), save the changes, and run the model.

The Dashboard Authoring Module is described in detail in the GoldSim Dashboard Authoring Module Users Guide.
Chapter 10: Advanced Modeling Concepts

These are much deeper waters than I had thought.

Sherlock Holmes (Sir Arthur Conan Doyle),
*The Reigate Squires*

Chapter Overview

This chapter describes a number of advanced and powerful GoldSim features. They have been saved for last since in order to use them, you must have a good understanding of the basic GoldSim features. Although you can build many kinds of models without using these features, once you become comfortable with GoldSim, many of these advanced features will seem indispensable, and you will want to take advantage of some of them in all of your models.

In this Chapter

The following topics are discussed in this chapter:

- Using Vectors and Matrices
- Understanding Locally Available Properties
- Solving Convolution Integrals
- Generating Stochastic Time Histories
- Using Resources
- Script Elements
- Using Conditional Containers
- Using External Application Elements
- Localizing Containers
- Cloning Elements
- Referencing an Output’s Previous Value
- Using Looping Containers
- Understanding the Causality Sequence
- Using SubModels to Embed Models Within Models
- Customized Importance Sampling Using User-Defined Realization Weights
- Dynamically Revising Distributions Using Simulated Bayesian Updating
- Tracking Model Changes
Using Vectors and Matrices

In many systems, you will want to create and manipulate elements that represent collections of data, rather than individual items. For example, you may want to create an element that represents your company’s revenue for each of the last five years, or an element that represents the salmon population in each of 20 streams.

One way to do this, of course, would be to create separate elements for each object you wanted to model (e.g., five elements representing revenue, 20 elements representing salmon populations).

Such an approach, however, is not desirable for two reasons:

- It could require you to create a very large number of elements (e.g., if you wanted revenues for the past 25 years or wanted to evaluate salmon populations in 200 streams). This could result in very large, cluttered models.
- Usually, you will want to carry out the same types of calculations and operations on all related objects in such a collection (e.g., multiply all revenues by 2, compute the number of salmon eggs this year for all 20 streams based on the current salmon population). Having to do this individually for every object in a large collection would be very cumbersome and time-consuming.

To address these kinds of problems, GoldSim allows you to create and manipulate vectors and matrices (collectively referred to as arrays). For example, you could create a vector element that represented the salmon populations in each of a number of streams:

You could also create a matrix element that represented the salmon population in each of streams over a period of three years:
In addition to adding data in the form of vectors and matrices, you can manipulate these arrays in equations. For example, you could create an Expression element, and define it as:

\[ 2 \times \text{Salmon\_Population} \]

The output of the Expression would be an array, identical to the Salmon\_Population array, except each item of the array would be two times greater.

If you did not use arrays to carry out this calculation, rather than creating 2 elements, you would need to create 40 elements (20 Data elements and 20 Expression elements) to accomplish the same thing!

Vectors and matrices are applicable to almost any kind of system, and you will likely want to take advantage of this feature in most of your models.

An example model which uses vectors and matrices within different elements (arrays.gsm) can be found in the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu). In addition, ArrayTimeSeries.gsm (in the Time Series subfolder of General Examples) includes examples of how Time Series elements can be used to generate arrays.

The following sections describe how you can create and use vectors and matrices in your models.

**Understanding Array Labels**

When you define a vector or a matrix, it must be defined relative to a particular set of array labels. A set of array labels is simply a collection of labels for the items of the array. The labels can be numbers (1, 2, ..., 10) or words (apples, oranges, peaches, pears). When you define an element which has an array as an output, you must specify the sets of array labels upon which the array is based.

You define and reference a particular item of an array by using these labels. Vectors require a single set of array labels, and matrices require two sets of array labels (one for the rows, one for the columns).

As an example, consider a matrix that represents the salmon population in each of a number of streams over a period of 3 years:
The matrix is defined relative to two sets of array labels: the rows are labeled using a "stream" set, and the columns are labeled using a "year" set. The labels or members for the "stream" set are BearCreek, IssaquahCreek, BlackRiver, and so on. The labels for the "year" set are the numbers 1998, 1999, and 2000.

You would reference the (scalar) item in the third row and second column of the matrix as Salmon_Population[BlackRiver, 1999].

Creating and Editing Array Labels

You can create (and edit) sets of array labels by selecting Model|Array Labels… from the main menu or by pressing the Array Labels button in the Advanced toolbar:

You can also create array label sets directly from the dialog for defining the Output Attributes for an element by selecting “Create new Array Labels…” from the drop-list:

Read more: Defining Vectors and Matrices Using Data Elements (page 790).

Regardless of how you do so, the following dialog for defining the array labels will be displayed:
For convenience, GoldSim provides two commonly-used sets of array labels: Days has seven items (the days of the week); and Months has twelve items (the months of the year). GoldSim extension modules may also add some special sets (e.g., the Contaminant Transport Module adds a Species and an Elements set).

To add a new set of array labels, press Add New Set…. A dialog for specifying the type of array labels is displayed:

You must select whether you wish to define a Named Set or an Indexed (numbered) Set. If you choose to create a Named Set of array labels (or select an existing Named set to be edited), a dialog for editing the set will be displayed:
You can assign a name to a new set of array labels (by default it will be named Setn). You add and remove entries using the Add and Remove buttons. Holding the Ctrl key down changes the Remove button to a Remove All button, so you delete all entries. You can also paste a column of labels (e.g., from a spreadsheet) within this dialog by pressing Ctrl+V.

After you add two or more entries, the Move Up and Move Down buttons become available, allowing you to move entries up or down in the list. If you reorder the items, any items in existing elements using the set will automatically be re-sorted to the new sequence.

If you choose to create an Indexed set of array labels (or select an existing Indexed set to be edited) a dialog for editing the set will be displayed.
You can assign a name to the new set of array labels (by default it will be named Setn).

You must then enter a **Start index** (e.g., 1) and an **End index** (e.g., 100). When you press the **Reapply** button, GoldSim will then create all the entries for the set.

If you are editing an existing indexed set which is already referenced by some elements for which data has been defined, and change the Start index, GoldSim will prompt you for how you would like to adjust the values in existing Data elements that reference that set.

**Read more:** Changing the Start Index for an Indexed Set that is Already Being Referenced (page 788).

The Array Label editing dialog also allows you to define the Styles that are used when an element based on the set is displayed in a result chart. The **Base Style**, in combination with the **Auto Colors**, **Vary Lines** and **Vary Symbols** options allow you to automatically generate the Styles by pressing the **Reapply** button. Alternatively, you can click on any Style in the list to manually specify the Style attributes:
Note: The Array Label Set dialog can also be accessed when viewing results (in order to access the Chart Style attributes). When accessed in this manner, in some cases, the Chart Style options may be hidden (if editing them is not applicable).

The **Show** checkbox determines whether or not the item is displayed in result charts.

**Read more:** [Viewing Time Histories for Array Outputs](#) (page 592); [Viewing a Vector Chart](#) (page 692); [Viewing a Matrix Chart](#) (page 694).

Note: Array label sets are saved with your model. If you wish to use the same set of array labels in different models, you must recreate it in each model. Alternatively, you can copy a set of array labels from one model file to another by copying an element that uses the set. Note, however, that GoldSim will only allow you to do so if a set of the same name does not exist in the model file into which the element is being copied.

To edit an existing set of array labels, select the set from the list in the Array Labels dialog and press **Edit Set**. The Array Label Set dialog will then appear for editing.

If you are editing an existing indexed set which is already referenced by some elements for which data has been defined, and change the Start index, GoldSim will prompt you for how you would like to adjust the values in existing Data elements that reference that set.

In particular, if the new range does not overlap with the old range, a dialog like this will appear to allow you to specify how the data is to be mapped to the new index values:
If you select Option 1, all data in existing Data elements will be cleared out (set to 0).

If you select Option 2, existing data will be mapped such that any data associated with the old Start Index will be mapped to the new Start Index, and so on. If the new range is shorter, data at the end of the range will be lost. If the new range is longer, zeros will be added to the new items.

If the new range does overlap with the old range, a slightly different dialog will be displayed for mapping the data:

If Option 1 is selected, GoldSim will only map entries for items in the old set which match items in the new set. For example, if the original set was 1 through 20, and the new (edited) set was 15 through 34, for any elements that referenced the set, only entries 15 through 20 would be mapped to the new set (as the first five entries), and entries associated with items 1 through 14 in the original set would be lost. The remaining entries associated with the new set (16 through 34) would all be set to zero.

If you select Option 2, existing data will be mapped such that any data associated with the old Start Index will be mapped to the new Start Index, and so on. If the new range is shorter, data at the end of the range will be lost. If the new range is longer, zeros will be added to the new items.

If you select Option 3, all data in existing Data elements will be cleared out (set to 0).

**Deleting a Set of Array Labels**

You can delete a set of array labels by selecting the set in the Array Labels dialog and pressing Delete Set....
Using Vectors and Matrices

Chapter 10: Advanced Modeling Concepts

GoldSim User’s Guide

Note, however, that this button will be grayed out if the set is being referenced somewhere in the model (you can not delete a set if it is referenced by an element). If a set is used in the model, the Show References… button will be available (otherwise, it is grayed out). Pressing this button displays a dialog that provides a list of all of the elements which reference the set, and allows you to edit or delete each element:

After selecting an element in this list, you can choose to either Delete or Edit it.

Once you have defined the sets of array labels that you wish to use, you can create vectors and matrices. The primary way to create vectors and matrices is to define them using Data elements. Once these exist, other vectors and matrices can be created by other kinds of elements based on these (e.g., by using these as inputs).

You specify that a Data element is to be a vector or a matrix when you define its output attributes (by pressing the Type… button in the element’s properties dialog). The dialog for specifying output attributes for an element looks like this:
The **Order** drop-list provides three choices: “Scalar” (the default), “Vector (1D-Array)”, and “Matrix (2D-Array)”. If you wish to create a vector, select “Vector (1D-Array)”. If you wish to create a matrix, select “Matrix (2D-Array)”. If you select “Vector (1D-Array)”, the **Row Labels** field becomes available. If you select “Matrix (2D-Array)”, both the **Row Labels** and **Column Labels** fields become available. These two fields contain lists of all of the sets of array labels defined in your model:

If you have defined the **Order** as a vector or a matrix (i.e., if the **Row Labels** or **Column Labels** fields are active), you must select a set of array labels (or GoldSim will display a warning message when you try to close the Output Attributes dialog).

Note that within the drop-list for the Row and Column labels is the choice to “Create new Array Labels…”. If you click this option, the dialog for defining new array labels will be displayed. After leaving the Array Labels dialog, you will be returned to the dialog for defining output attributes.

**Read more:** [Creating and Editing Array Labels](#) (page 784).

To illustrate how you would create a vector or a matrix Data element, consider the following example. Suppose that you defined the **Order** as “Vector (1D-Array)”, and defined the **Row Labels** as “Days”. The properties dialog for the Data element would look like this.
Note that this dialog is different from the standard Data element dialog for a scalar.

Read more: Data Elements (page 151).

In particular, because it is an array, you cannot directly edit the Data input field. To edit the vector, you must press Edit Data…, which will display the following dialog.

This dialog displays all items of the vector (in this case, seven). Each field is a standard edit field (you can enter a number, an expression or a link), although typically you would enter just a number (with units).

Note that the equivalent dialog for a matrix looks like this (in this example using a set of array labels for the Column Labels named Fruit with members Apples, Oranges, Peaches and Pears):
Although you would typically enter constants (numbers) into the input fields defining a vector or a matrix, the fields accept links and expressions. For example, you could create a vector of Stochastics by entering a link to a Stochastic for each item in a vector.

Note: You can paste data from a spreadsheet, a Word table, or a tab-delimited text file directly into a vector or matrix grid. To do so, copy the data to the clipboard, select the cell in the vector or matrix representing the upper left-hand corner of the range into which you wish to paste (by single-clicking in it), and press Ctrl+V.

Note: You can resize the columns of the editing dialog by dragging the lines separating columns right or left. You can also resize the entire dialog by dragging a side (or a corner).

**Viewing an Array in a Browser or Interface**

When you view a vector or a matrix in a browser or an interface, the inputs and outputs can be expanded to show the individual items.

Note: Element inputs and outputs are only shown in the browser if you choose to **Show element subitems** (accessed via the browser context menu by right-clicking in the browser).

Note that placing the cursor over an item of the array displays the **Current Value** (if in Edit Mode) or the **Last Value** (if in Result Mode). The **Current Value** is computed as the expected value of the item. The Last Value is the final value (i.e., the value at the end) of the last realization.

Read more: [Understanding Result Mode](#) (page 556).

**Referencing an Item of an Array**

If required, you can access a particular item of the array in an expression. You do this by using brackets [ ].

For example,

    Data1[Monday]

references a single (scalar) value representing the second item of the vector Data1 (defined by the set “Days”, which begins with Sunday).

Similarly,

    Data2[Monday, Peaches]
Using Vectors and Matrices

references a single (scalar) value representing the item in the second row and third column of the matrix Data2 (with the rows defined by the set “Days”, and the columns defined by the set “Fruit”, which consists of Apples, Oranges and Peaches).

When referencing array items, you can also use variables as the arguments. For example, if Data1 was a vector of “Days”, and X was a scalar value, you could write the following in an expression:

Data1[X]

If X was 3, it would return the 3rd item in the vector (i.e., Data1[Tuesday]).

GoldSim rounds to the nearest integer when evaluating an array argument, so in this example, as long as the rounded value of X was between 1 and 7 inclusive, GoldSim would be able to evaluate the expression. If X evaluated to a number outside of that range, GoldSim would display a fatal error. Of course, X could change with time (or any other variable in your model).

This functionality can also be used with matrices. As an example, if Data2 was a matrix with the rows defined by the set “Days”, and the columns defined by the set “Fruit” (with 3 entries), you could reference the following in an expression:

Data2[X, Y]

In this case, X would need to be an integer between 1 and 7, inclusive, and Y would need to be an integer between 1 and 3 inclusive. If X was 3 and Y was 2, it would return the item in row 3, column 2 of the matrix (i.e., Data2[Tuesday, Peaches] assuming Peaches was the second array label for “Fruit”).

When using variables to reference array items, the following points should be noted:

- If the array is based on a named set, the variable represents the ordinal item in the set (i.e., the row or column number). For example, if Data1 was vector of “Days”, and X was 3, then Data1[X] would return the 3rd item in the vector (i.e., Data1[Tuesday]).

- If the array is based on an indexed set, the variable represents the actual item with that number label in the set. For examples, if Data9 was a vector defined by an indexed set with labels 3, 4, 5 and 6, and X was 3, then Data9[X] would return the first item in the vector (i.e., Data9[3]).

Finally, when using this with matrices, you can use * as a wildcard. Using the example above,

Data2[*, Y]

would return column Y (a vector of Days). Similarly,

Data2[X, *]

would return row X (a vector of Fruit).

This ability to use variables when referencing array items is particularly powerful when using this feature in conjunction with array constructor functions.

Read more: Defining Arrays in an Input Field Using Array Constructor Functions (page 795).
In addition to using Data elements to create vectors, you can also create vectors using Stochastic elements. If you define a Stochastic element as a vector, rather than inputting a single probability distribution, you specify a set of probability distributions (one for each item of the vector).

If you specify the Stochastic as a vector (by specifying a set of array labels via the Type… button), the dialog for defining the distribution (accessed via the Edit Distribution… button) will look similar to this:

All the inputs to the Stochastic must be vectors.

**Read more:** Creating a Stochastic Vector (page 183).

One of the quickest ways to create an array is to use array constructor functions. These are functions that can be entered into an input field that generate an array (a vector or a matrix) by specifying the array label set(s) and values as arguments.

For example, the following expression generates a vector based on the Days array label set, in which all the items have a value of 1m:

```
Vector(days, 1m)
```

The following expression generates a matrix based on the Days array label set (for rows) and the Months array label set (for columns), in which all the items have a value of 0 g:

```
Matrix(days, months, 0g)
```

**Note:** You do not have to specify the array label set(s) when using an array constructor. If you omit them, GoldSim will assume that the constructed array has the same order and set(s) as that of the input field where it is being defined. For example, if you created an Expression defined as a vector with array label set “days”, then to create a vector of zeros, you could enter either “Vector(days, 0)” or simply “Vector(0)”.

---

**Defining Vectors Using Stochastic Elements**

**Defining Arrays in an Input Field Using Array Constructor Functions**
In addition to creating an array in which all the values for the items are identical (and hence specified by a single argument), you can also specify each item separately:

Vector(days, 1m, 3m, 0m, 5m, 6m, 2m, 7m)

In this example, since the Days array label set was specified, seven items are defined.

**Note:** When defining a matrix using an array constructor, the first argument represents the row array label set (having n items), the second argument is the column label set (having m items), and if all of the values are specified separately, they are defined as follows: \( r_1c_1, r_1c_2, \ldots, r_1c_m, r_2c_1, \ldots, r_nc_m \). That is, the items are listed left-to-right-then-down through the matrix.

The units for the values of a constructor do not need to be identical. The following expression is valid:

Vector(days, 1m, 3ft, 0km, 5mm, 6m, 2ft, 7m)

If any of the values listed do not have units, they take on the units of the first item in the array:

Vector(days, 1m, 3ft, 0km, 5mm, 6m, 2, 7m)

In the example above, the sixth item of the vector would be 2m.

Vector(days, 1m3/day, 3, 0, 5, 6, 2, 7)

In the example above, all items have units of m3/day.

The values for the items of an array can be specified as links. For example, if X was an Expression with dimensions of volume/time, it could be inserted into the example above:

Vector(days, X)

In this case all 7 items would take on the value of X. Note that links such as X do not have to represent constants. They can be functions of time.

In addition, when using the Matrix constructor, you can specify vector arguments. That is, you can use the Matrix constructor to build a matrix from specified vectors. The rules for doing so are as follows:

- If you create a matrix that has a different number of rows than columns, you can define the matrix by providing vectors for either all of the rows, or all of the columns. GoldSim then intelligently creates the matrix in the appropriate way (i.e., it automatically determines based on the array label sets whether the vectors should be treated as rows or columns).

- If the matrix is square and the array label set of the provided vector matches the column array label set of the matrix, the vector values are loaded into the matrix by row. Under all other circumstances, the vector values are loaded into the matrix by column.

This can best be illustrated through some examples. These examples utilize array label sets called V2 with two items and V3 with three items, and vectors X2a, X2b, X2c, X3a and X3b defined below (based on the V2 and V3 array label sets):

X3a = [1, 2, 3]
X3b = [6, 7, 8]
X2a = [10, 20]
X2b = [12, 22]
X2c = [30, 40]

<table>
<thead>
<tr>
<th>Expression</th>
<th>Resulting Array</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix(V2, V3, X2a, X2b, X2c)</td>
<td>10  12  30</td>
</tr>
<tr>
<td></td>
<td>20  22  40</td>
</tr>
<tr>
<td>Matrix(V2, V3, X3a, X3b)</td>
<td>1   2   3</td>
</tr>
<tr>
<td></td>
<td>6   7   8</td>
</tr>
<tr>
<td>Matrix(V2, V2, X2b, X2c)</td>
<td>12  22</td>
</tr>
<tr>
<td></td>
<td>30  40</td>
</tr>
</tbody>
</table>

When using array constructors, you can also utilize two specialized variables: row and col. These variables can be used to define values based on the row and the column being referenced. Within constructors, these variables are defined as follows:

- If the array label set representing the row variable is based on a named set, row represents the row number. Likewise, if the array label set representing the column variable is based on a named set, col represents the column number.

- If the array label set representing the row variable is an indexed set, row represents the actual item label. Likewise, if the array label set representing the column variable is an indexed set, col also represents the actual item label.

They are best illustrated through some examples. These examples utilize a named array label set called Set1 with 3 items (A, B and C), an indexed array label set called Set2 with 4 items (1,2,3 and 4), and an indexed array label set called Set3 with 3 items (3,4,5).

<table>
<thead>
<tr>
<th>Expression</th>
<th>Resulting Array</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector(Set1,row)</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Vector(Set3,row)</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td>Vector(Set1, if(row&gt;=2,1,0))</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Vector(Set3, if(row&gt;=2,1,0))</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
### Using Vectors and Matrices

<table>
<thead>
<tr>
<th>Expression</th>
<th>Resulting Array</th>
</tr>
</thead>
</table>
| Matrix(Set1, Set2, row) | 1 1 1 1  
2 2 2 2  
3 3 3 3 |
| Matrix(Set1, Set2, col) | 1 2 3 4  
1 2 3 4  
1 2 3 4 |
| Matrix(Set2, Set2, if(col>=row,1,0) | 1 1 1 1  
0 1 1 1  
0 0 1 1  
0 0 0 1 |

One powerful use of array constructors is to take advantage of the ability of arrays to reference variables as arguments. For example,

Vector(days, if(row<=4, A[row], B[row]))

would create a vector of days in which the first 4 items would be drawn from the vector A, and the last 3 items would be drawn from the vector B.

**Read more:** Referencing an Item of an Array (page 793).

**Warning:** If you have elements with the names Row or Col, GoldSim will link to the element rather than use the keyword. Therefore, if you want to use these keywords, you should ensure that no elements exist in your model with these names.

---

### Using a Vector as a Lookup Table

Two array functions are available in GoldSim whose primary application is to allow you to create dynamic lookup tables (in which the values for the independent and/or dependent variables are changing with time).

Although GoldSim provides a specialized Lookup Table element, this element does not support dynamic tables (all the values must be constant values).

**Read more:** Using Lookup Table Elements (page 294).

Using the array functions vIndex and vInterp provides a mechanism for creating a dynamic lookup table using a vector.

**vIndex(vector, value)** requires the first argument to be a vector, and the second argument to be a value (with the same dimensions as the vector). It searches for the specified value within the provided vector, and returns the dimensionless index at which the value is found. It treats the vector as a continuous function, linearly interpolating where necessary between the defined points in the vector, and as a result, the returned index value is not necessarily an integer. For example, if the vector’s values were [3m, 4m, 5m, 6m], then vIndex(vector, 5.1m) would return an index value of 3.1. Similarly, vIndex(vector, 10ft) would return an index value of 1.048. If the array labels are indexed, the appropriate index value is returned; if the array labels are named, the first entry in the array has an index value of 1.

If the value is outside of the range of values in the vector, a fatal error is displayed. If the value exists multiple times in the vector (e.g., a sine function), the first matching value is used.
Using Vectors and Matrices

vInterp(vector, index) requires the first argument to be a vector, and the second argument to be a dimensionless index. It interpolates into the vector using the specified index value. It treats the vector as a continuous function, linearly interpolating where necessary between the defined values in the vector, and returns a value (with the same dimensions as the vector). For example, if the vector’s values were [3m, 4m, 5m, 6m] then vInterp(vector, 3.1) would return a value of 5.1m.

If the index is outside of the range of indices for the vector, a fatal error is displayed.

These two functions can be used in conjunction to create a dynamic lookup table. To do so, you would create two vectors using the same array label set, which would typically be indexed (e.g., 1 through 10). One of the vectors (named VectorInd here) would contain the independent variables, and the other vector (named VectorDep here) would contain the corresponding dependent variables. The values of these could change dynamically. At any time, if you wanted to obtain the dependent variable for a particular value of the independent variable (say, X), you could do so using the following expression:

\[ \text{vInterp(VectorDep, vIndex(VectorInd, X))} \]

This would look up the index of X in the vector representing the independent variable and return the corresponding value from the vector representing the dependent variable.

A reverse lookup into the same table could be carried out using this expression:

\[ \text{vInterp(VectorInd, vIndex(VectorDep, Y))} \]

This would look up the location of Y in the vector representing the dependent variable and return the corresponding value from the vector representing the independent variable.

An example model which illustrates how these functions can be used to create dynamic lookup tables DynamicLookup.gsm) can be found in the General Examples/LookupTable folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

Although you often will use Data elements to create arrays in your models, the real power of arrays is that they can be manipulated using other elements. In general, when you manipulate arrays in other elements, GoldSim carries out the calculations in parallel for each item of the array. GoldSim also provides a wide variety of special array functions which allow you to manipulate arrays.

In this section, the various operators and functions that can be used to build expressions involving arrays are discussed. The manner in which various elements can manipulate array inputs is also described.

The manner in which the standard operators in GoldSim can be used with arrays is summarized below:

**Addition and Subtraction of Arrays.** The addition (+) and subtraction (-) operators can be used between arrays if and only if the two arrays have the same dimensions and order, and are defined using the same set of array labels. GoldSim carries out the operation term-by-term and produces an array of the same order. For example, if A was a vector based on the set “Days”, and B was a vector based on the set “Days”, C=A + B would also produce a vector based on the set “Days”. The first item of C would be the sum of the first item of A and the first item of B; the second item of C would be the sum of the second item of A and the second item of B, and so on. You cannot add or subtract an array to/from a scalar.
Multiplication or Division of an Array by a Scalar. The multiplication (*) and division (/) operators can be used between arrays and a scalar. Each item of the array is multiplied or divided by the scalar. For example, you could create the expression 2 * A, where A was a vector. The output of the Expression would be a vector, identical to the A vector (and its output attributes would need to be defined accordingly), except each item of the vector would be two times as great. Note that not only can you divide an array by a scalar, but you can also divide a scalar by an array. In both cases, the result is an array.

Multiplication and Division of Arrays of Same Order. When the multiplication (*) and division (/) operators are used between arrays that have the same order (and are defined using the same set of array labels), GoldSim carries out the operation term-by-term and produces an array of the same order as the original arrays. For example, if A was a vector based on the set “Days”, and B was a vector based on the set “Days”, C=A * B would also produce a vector based on the set “Days”. The first item of C would be the product of the first item of A and the first item of B, etc.

Multiplying and/or Dividing the Rows or Columns of a 2-D Array (Matrix) by a 1-D Array (Vector). The following operations are supported in GoldSim for manipulating matrices: Matrix*Vector, Vector*Matrix, and Matrix/Vector. In these cases, the array label set of the Vector must match the array label set of either the rows or columns for the Matrix. If it matches that of the rows, then each row of the matrix is multiplied (or divided) by the corresponding term in the Vector. If it matches that of the columns, then each column of the matrix is multiplied (or divided) by the corresponding term in the Vector. If it matches both, the operation is carried out on the rows. In either case, the result is a Matrix of the same order as the original matrix.

Note: The operation described here is quite different from the linear algebra operation of multiplying a Vector by a Matrix (or vice versa), with the result being a Vector. That particular operation can be carried out using the specialized array function “Mult”.

Read more: Array Functions (page 801).

Raising an Array to a Power. You can use the exponentiation operator (** or ^) to raise the items of an array to a power. GoldSim carries out the operation term-by-term and produces an array of the same order. For example, you could create an expression named C defined as A^2, where A was a vector. C must have the same order and set of array labels as A. The first item of C would be the first item of A squared, the second item of C would be the first item of A squared, etc.

Using Relational Operators with Arrays. Relational operators (e.g., >, <, =, ==, >=) can be used between arrays if and only if the two arrays have the same dimensions and order, and are defined using the same set of array labels. GoldSim carries out the operation term-by-term and produces an array of conditions. For example, if A was a vector based on the set “Days”, and B was a vector based on the set “Days”, the Expression C defined as A > B would produce a vector based on the set “Days”. The first item of C would be the outcome (true or false) of the expression “first item of A” > “first item of B”; the second item of C would be the outcome (true or false) of the expression “second item of A” > “second item of B”, and so on.
Relational operators can also be used between arrays and scalars. For example, if A was a vector based on the set “Days”, and W was a scalar, the Expression D defined as A > W would produce a vector based on the set “Days”. The first item of D would be the outcome (true or false) of the expression “first item of A” > ”scalar value W”; the second item of D would be the outcome (true or false) of the expression “second item of A” >”scalar value W”, and so on.

Using Arrays in If Statements. If statements can use mixtures of arrays and scalars. The rules for how such statements are interpreted are as follows:

1. The first argument (the condition) can be an array or scalar. If it is an array, then the second and third argument can either be arrays with the same set of array labels as the condition or scalars. Scalars are treated as arrays of identical values with the same set of array labels as the condition. If the first argument is an array, GoldSim does an item by item evaluation to construct the output array.

2. If the condition is scalar, and the second and third arguments are arrays, they must be of the same order, and the output has the same order as these arguments. Either one array or the other is output in its entirety as the result.

3. If the condition is a scalar, one of the latter two arguments can be a scalar, and one can be an array. The output of the If statement is then an array, and the scalar is treated as an array of identical values. Either one array or the other is output in its entirety as the result.

Read more: Entering and Editing Expressions in Input Fields (page 81).

Array Functions

Most of the built-in functions in GoldSim can also operate on arrays. In most cases, these functions carry out the operation term-by-term and produce an array of the same order as the input argument. For example, if A was a vector based on the set “Days”, C=sin(A) would also produce a vector based on the set “Days”. The first item of C would be the sine of the first item of A, the second item of C would be the sine of the second item of A, and so on.

Several functions (max, min, mod, tdist, tprob, bess, beta) accept multiple array arguments. For these functions, the arguments must all be scalars, or must be arrays of the same order that are defined using the same set of array labels. GoldSim carries out the operation term-by-term and produces an array of the same order. For example, if A was a vector based on the set “Days”, and B was a vector based on the set “Days”, C=min(A, B) would also produce a vector based on the set “Days”. The first item of C would be the minimum of the first item of A and the first item of B; the second item of C would be the minimum of the second item of A and the second item of B, and so on.

Two specialized functions (occurs and changed) cannot accept arrays as arguments.

If X and Y are arrays (of the same order and using the same set of array labels), the If(C, X, Y) function can manipulate the arrays in two different ways:

- If C is a scalar condition, the output of the If function is either the array X (if C is true) or the array Y (if C is false).
- If C is an array (of the same order and using the same set of array labels as X and Y), GoldSim will carry out a term-by-term if,then computation to produce the output array. That is, if the first item of C
is true, the first item of the output would be equal to the first item of X, otherwise it would be equal to the first item of Y, and so on.

Read more: Built-in Functions (page 124).

In addition to the standard functions mentioned above, GoldSim also provides a wide variety of special array functions which allow you to manipulate arrays (accessible from the context menu for an input field).

These special array functions are listed below:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetItem(VC,n)</td>
<td>Returns the nth item in the vector.</td>
<td>scalar</td>
</tr>
<tr>
<td>GetItem(MC,n, m)</td>
<td>Returns the item in the nth row and mth column of the matrix.</td>
<td>scalar</td>
</tr>
<tr>
<td>GetRow(VC,n)</td>
<td>Returns the nth item in the vector. (If first argument is a vector, this function is identical to GetItem).</td>
<td>scalar</td>
</tr>
<tr>
<td>GetRow(MC,n)</td>
<td>Returns the nth row of the matrix.</td>
<td>vector</td>
</tr>
<tr>
<td>GetColumn(MC,m)</td>
<td>Returns the mth column of the matrix.</td>
<td>vector</td>
</tr>
<tr>
<td>GetRowCount(VC)</td>
<td>Returns the number of items in the vector.</td>
<td>scalar</td>
</tr>
<tr>
<td>GetRowCount(MC)</td>
<td>Returns the number of rows in the matrix.</td>
<td>scalar</td>
</tr>
<tr>
<td>GetColumnCount(MC)</td>
<td>Returns the number of columns in the matrix.</td>
<td>scalar</td>
</tr>
<tr>
<td>sumv(VC, X)</td>
<td>Sums the items of the vector. If the arguments are conditions, ORs the conditions. X is optional. If positive, the calculation is applied to the first X terms; if negative, the calculation applies to the last X terms. X is ignored if 0 or greater than array size.</td>
<td>scalar</td>
</tr>
<tr>
<td>prodv(VC, X)</td>
<td>Computes the product of the items of the vector. If the arguments are conditions, ANDs the conditions. X is optional. If positive, the calculation is applied to the first X terms; if negative, the calculation applies to the last X terms. X is ignored if 0 or greater than array size.</td>
<td>scalar</td>
</tr>
<tr>
<td>minv(V)</td>
<td>Computes the smallest item in the vector.</td>
<td>scalar</td>
</tr>
<tr>
<td>maxv(V)</td>
<td>Computes the largest item in the vector.</td>
<td>scalar</td>
</tr>
<tr>
<td>meanv(V)</td>
<td>Computes the mean of the items in the vector.</td>
<td>scalar</td>
</tr>
<tr>
<td>sdev(V)</td>
<td>Computes the standard deviation of the items in the vector.</td>
<td>scalar</td>
</tr>
<tr>
<td>rowmin(V)</td>
<td>Computes the index of the smallest item in the vector (i.e., 1, 2, 3, etc.).</td>
<td>scalar</td>
</tr>
<tr>
<td>rowmax(V)</td>
<td>Computes the index of the largest item in the vector (i.e., 1, 2, 3, etc.).</td>
<td>scalar</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>Result</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>sort123(V)</td>
<td>Sorts the items in the vector from smallest to largest.</td>
<td>vector</td>
</tr>
<tr>
<td>sort321(V)</td>
<td>Sorts the items in the vector from largest to smallest.</td>
<td>vector</td>
</tr>
<tr>
<td>dot(V1,V2)</td>
<td>Dot (inner) product of V1 and V2. The first vector is assumed to be a row vector and the second vector is assumed to be a column vector. The two vectors must be based on the same set of array labels.</td>
<td>scalar</td>
</tr>
<tr>
<td>vvmatrix(V1,V2)</td>
<td>Vector multiplication (to produce a matrix). The first vector is assumed to be a column vector and the second vector is assumed to be a row vector. This is equivalent to vector multiplication of V1 by the transpose of V2.</td>
<td>matrix[A,B], where A is the set of array labels for V1 and B is the set of array labels for V2</td>
</tr>
<tr>
<td>vIndex(V1,X)</td>
<td>Vector lookup. The second argument must be a value (with the same dimensions as the vector). It searches for the specified value within the provided vector, and returns a dimensionless index at which the value is found. It treats the vector as a continuous function, linearly interpolating where necessary between the defined points in the vector.</td>
<td>scalar</td>
</tr>
<tr>
<td>vInterp(V1,n)</td>
<td>Vector interpolation. The second argument must be a dimensionless index. It interpolates into the vector using the specified index value. It treats the vector as a continuous function, linearly interpolating where necessary between the defined values in the vector, and returns a value (with the same dimensions as the vector).</td>
<td>scalar</td>
</tr>
<tr>
<td>sumr(MC, X)</td>
<td>Sums the items across each row of the matrix. If the arguments are conditions, ORs the conditions. X is optional. If positive, the calculation is applied to the first X terms; if negative, the calculation applies to the last X terms. X is ignored if 0 or greater than array size.</td>
<td>vector[A], where A is the set of array labels for the rows in the original matrix</td>
</tr>
<tr>
<td>prodr(MC, X)</td>
<td>Computes the product of the items across each row of the matrix. If arguments are conditions, ANDs the conditions. X is optional. If positive, the calculation is applied to the first X terms; if negative, the calculation applies to the last X terms. X is ignored if 0 or greater than array size.</td>
<td>vector[A], where A is the set of array labels for the rows in the original matrix</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>Result</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------</td>
</tr>
<tr>
<td>minr(M)</td>
<td>Computes the smallest item in each row of the matrix.</td>
<td>vector[A], where A is the set of array labels for the rows in the original matrix</td>
</tr>
<tr>
<td>maxr(M)</td>
<td>Computes the largest item in each row of the matrix.</td>
<td>vector[A], where A is the set of array labels for the rows in the original matrix</td>
</tr>
<tr>
<td>meanr(M)</td>
<td>Computes the mean of the items in each row of the matrix.</td>
<td>vector[A], where A is the set of array labels for the rows in the original matrix</td>
</tr>
<tr>
<td>sdr(M)</td>
<td>Computes the standard deviation of the items in each row of the matrix.</td>
<td>vector[A], where A is the set of array labels for the rows in the original matrix</td>
</tr>
<tr>
<td>sumc(MC, X)</td>
<td>Sums the items down each column of the matrix. If the arguments are conditions, ORs the conditions. X is optional. If positive, the calculation is applied to the first X terms; if negative, the calculation applies to the last X terms. X is ignored if 0 or greater than array size.</td>
<td>vector[B], where B is the set of array labels for the columns in the original matrix</td>
</tr>
<tr>
<td>prodc(MC, X)</td>
<td>Computes the product of the items down each column of the matrix. If the arguments are conditions, ANDs the conditions. X is optional. If positive, the calculation is applied to the first X terms; if negative, the calculation applies to the last X terms. X is ignored if 0 or greater than array size.</td>
<td>vector[B], where B is the set of array labels for the columns in the original matrix</td>
</tr>
<tr>
<td>minc(M)</td>
<td>Computes the smallest item in each column of the matrix.</td>
<td>vector[B], where B is the set of array labels for the columns in the original matrix</td>
</tr>
<tr>
<td>maxc(M)</td>
<td>Computes the largest item in each column of the matrix.</td>
<td>vector[B], where B is the set of array labels for the columns in the original matrix</td>
</tr>
<tr>
<td>meanc(M)</td>
<td>Computes the mean of the items in each column of the matrix.</td>
<td>vector[B], where B is the set of array labels for the columns in the original matrix</td>
</tr>
</tbody>
</table>
### Function Description

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>sdc(M)</td>
<td>Computes the standard deviation of the items in each column of the matrix.</td>
<td>vector[B], where B is the set of array labels for the columns in the original matrix</td>
</tr>
<tr>
<td>trans(MC)</td>
<td>Transpose of matrix. trans(MC1[A,B]) = MC2[B,A]</td>
<td>matrix[B,A], where A is the set of array labels for the rows in the original matrix and B is the set of array labels for the columns</td>
</tr>
<tr>
<td>inv(M)</td>
<td>Inverse of matrix. inv(M1[A,B]) = M2[B,A]</td>
<td>matrix[B,A], where A is the set of array labels for the rows in the original matrix and B is the set of array labels for the columns</td>
</tr>
<tr>
<td>mult(M1,M2)</td>
<td>Matrix multiplication. The columns of the first matrix must be based on the same set of array labels as the rows of the second matrix. Mult(M1[A,B] , M2[B,C]) = M3[A,C].</td>
<td>matrix[A,C], where A is the set of array labels for the rows in matrix M1 and C is the set of array labels for the columns in matrix M2</td>
</tr>
<tr>
<td>mult(M,V)</td>
<td>Matrix times vector. The vector is automatically treated as a column vector. The vector must be based on the same set of array labels as the columns of the matrix. Mult(M[A,B] , V[B]) = V[A]</td>
<td>vector[A], where A is the set of array labels for the rows in matrix M</td>
</tr>
<tr>
<td>mult(V,M)</td>
<td>Vector times matrix. The vector is automatically treated as a row vector. The vector must be based on the same set of array labels as the rows of the matrix. Mult(V[A] , M[A,B]) = V[B]</td>
<td>vector[B], where B is the set of array labels for the columns in matrix M</td>
</tr>
</tbody>
</table>

V, V1, V2: Vector of values  
VC: Vector. Can be value or condition.  
M, M1, M2: Matrix of values  
MC, MC1, MC2: Matrix. Can be value or condition.  
X: Scalar value.  
n,m: a dimensionless value; can be specified as a number, equation or a link. Real values are rounded to the nearest integer.

Note that the GetItem, GetRow, and GetColumn functions can, in most cases, more easily be implemented by referencing items of an array directly using variables. That is, if A was based on a named set or an indexed set that started with 1, GetItem(A, X) is identical to A[X].

Note, however, that in his example, if A was an indexed set that did not start with 1, GetItem(A, X) would not be identical to A[X]. GetItem(A, X) would return the Xth row. A[X] would return the item whose label was X. These are only the same if the indexed set starts with 1.

Another difference is that the first argument of the GetItem, GetRow and GetColumn functions can itself be an expression. To implement this using...
arrays with variable arguments would therefore require you to create a second element. That is, to reproduce this expression:

\[ \text{GetItem}(A \ast B, X), \]

you would first need to create a vector \( C = A \ast B \), and then reference \( C[X] \).

Read more: Referencing an Item of an Array (page 793).

Many of the GoldSim elements can manipulate and produce vectors and matrices. For example, you can sum a number of matrices using the Sum element, or compute the peak value of each item of a vector using the Extrema element.

In general, if the outputs for an element are an array, its inputs must be arrays (defined by the same sets of array labels), and GoldSim carries out the element’s calculations in parallel for each item of the array.

Note: If an input field of an element requires a vector, and the vector only has a single item, GoldSim allows you to enter a scalar as the input. Of course, if you then add items to the set of array labels (such that it consists of more than one item), the scalar input will no longer be valid.

In order for an element to output an array, you must specify that it is to be a vector or a matrix when you define its output attributes (by pressing the Type... button in the element’s properties dialog). The dialog for specifying output attributes for an element looks like this:

The Order drop-list provides three choices: “Scalar” (the default), “Vector (1D-Array)”, and Matrix (2D-Array)”: If you wish to create a vector, select “Vector (1D-Array)”. If you wish to create a matrix, select “Matrix (2D-Array)”. If you select “Vector (1D-Array)”, the Row Labels field becomes available. If you select “Matrix (2D-Array)”, both the Row Labels and Column Labels fields become available.

These two fields contain lists of all of the sets of array labels defined in your model:
If you have defined the **Order** as a vector or a matrix (i.e., if the **Row Labels** or **Column Labels** fields are active), you must select a set of array labels (or GoldSim will display a warning message when you try to close the Output Attributes dialog):

Within the drop-list for the Row and Column labels is the choice to “Create new Array Labels…”. If you click this option, the dialog for defining new array labels will be displayed. After leaving the Array Labels dialog, you will be returned to the dialog for defining output attributes.

**Read more:** [Creating and Editing Array Labels](page 784).

**Note:** Element inputs and outputs are only shown in the browser if you choose to **Show element subitems** (accessed via the browser context menu by right-clicking in the browser).

You can also view array output results in table form or chart form (as bar charts).

**Read more:** [Viewing Array Results](page 689).

**Copying Array Elements Between Models**

When you copy an element defined using a set of array labels (i.e., an element with vector or matrix outputs) to another model, you must ensure that the set(s) of array labels referenced by the element in the two models are consistent. In particular, GoldSim imposes the following rules:

- If the set(s) of array labels referenced by the element already exist in the model into which it is being pasted, the number of items in the set(s) must be identical in both models. Note that it is not necessary that the set labels be identical, as long as the number of items is identical.
If the set(s) of array labels referenced by the element do not exist in the model into which it is being pasted, GoldSim will automatically create them in the new model.

Read more: Copying Elements Between Model Files (page 103).

Understanding Locally Available Properties

Locally available properties are special attributes of some elements in GoldSim. Locally available properties are similar to element outputs in that they have a data type (e.g., value, condition), order (e.g., scalar, vector) and dimensions (i.e., units). However, they do not appear as outputs of the element to which they belong. Rather, they are only visible in browsers (the main browser, or the Insert Link browser).

Note: Locally available properties are only shown in the main browser if you choose to Show element subitems (accessed via the browser context menu by right-clicking in the browser).

Locally available properties derive their name from the fact that they may only be available, or they may take on different values (i.e., be over-ridden), in “locally available” parts of your model (e.g., within particular Containers).

Containers are often providers of locally available properties. In fact, the Run Properties are actually locally available properties belonging to the Model Container:

Read more: Understanding and Referencing Run Properties (page 486).

These properties are “broadcast” to the entire model. A Run Property (like ETime) is an example of a type of locally available property that is available throughout a model, but can take on different values in different parts of the model. In particular, some of the Run Properties can be over-ridden locally if a Container is defined to have an internal clock (a local timestep).

Read more: Specifying Containers with Internal Clocks (page 474).

In such a case, the Container with the internal clock also possesses Run Properties:
If you define an internal clock for a Container, and then within that Container, reference ETime, GoldSim will automatically connect to the “locally available” ETime (the ETime in the Container), rather than the global ETime (the ETime associated with the “root” or Model Container). Hence, the actual locally available property that it linked to is a function of where it is referenced from.

Some locally available properties are only available in specific locations. A good example of this is the loop count for a Looping Container.

Read more: Using Looping Containers (page 963).

Within the looping Container itself (or in its property dialog), it is often necessary to reference the loop count. However, this variable has no meaning whatsoever outside of the looping Container. As a result, it can only be referenced inside the Container (including in its property dialog). If you right-click in an input field within the looping Container and select Insert Link to view the Insert Link browser, the loop count will appear in an Available Properties folder:

If you invoke the Insert Link browser from outside of the Container, however, the loop count is not visible and cannot be referenced.

Locally available properties are referenced in expression by adding the ~ prefix. For example, when you insert the loop count for a looping Container into a field, it appears as follows:
Modeling Aging Chains

In some situations, it is necessary to keep track of the age structure of a stock of material or objects. For example, you may want to track the number of people in each of a number of age groups, the number of people in a company at different experience levels (e.g., new hire, experienced, expert), or the number of trucks of different age groups on the road.

To model such a situation, you cannot use a single Stock (e.g., a Reservoir). Rather, you must disaggregate the total stock into multiple categories (referred to as cohorts). Each cohort “graduates” to the next cohort over time (and can only move in one direction). However, each cohort may grow and shrink for other reasons (e.g., if you were modeling a population of people, a particular category could grow due to immigration, and shrink due to emigration and death). Note that these rates of growing and shrinking are likely a function of the specific cohort (e.g., death rates).

Depending on your conceptual model, there are three fundamental ways to model such a chain:

- Using a series of Reservoirs or Pools.
- Using a series of Material Delays.
- Using a series of Integrators.

These three methods are discussed briefly in the following sections.

An example file which illustrates how aging chains can be modeled (AgingChain.gsm) can be found in the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

Perhaps the simplest way of modeling aging chains is to create a series of Reservoirs (or Pools; the example here uses Reservoirs). The transition rate from one cohort (say C2) to the next, is set equal to C2/Residence_Time (a first-order rate), where C2 is the quantity in cohort 2, and Residence_Time is the average residence time in cohort 2.

Read more: Reservoir Elements (page 235); Pool Elements (page 252).

In such a situation, the Reservoir for cohort 2 would look like this:
In this example, the cohort represents people aged 1 to 2 years. Hence, the Residence_Time is equal to 1 year. The rate of addition is the withdrawal rate from the previous cohort (C1), and the rate of withdrawal is C2/Residence_Time. Note that this simple representation does not represent any losses (e.g., due to death). To do so, you would simply need to add an additional withdrawal (which typically would be some fraction of C2).

It is important to understand that due to the nature of a Reservoir, such a representation results in a behavior which is typically inappropriate for modeling specific age groups. To illustrate this, let’s assume 1) that cohort 2 starts with 1000 one-year olds; 2) there is no addition to the cohort (C1 has an initial value of 0), and 3) there is no death rate. Cohort 2 only decreases due to graduations to the next cohort. Under these circumstances, the number of people in cohort 2 would look like this:
As can be seen, rather than staying at 1000 for the first year, and then abruptly changing to 0 (when all of the one-year olds actually become 2 and hence should immediately graduate to the next cohort), the number of one-year olds gradually decays over a period of 5 years. As a result, modeling an aging chain in this manner is most appropriate when the residence time in the cohort represents an average time, with some people leaving earlier and some later. This is the case, for example, if the cohort represents a group of employees (e.g., inexperienced), rather than a specific age group (e.g., between 1 and 2 years old). One of the other two methods of modeling aging chains (using Material Delay or Integrators with discrete pushes) should generally be used when modeling specific age groups.

An example file which illustrates how aging chains can be modeled (AgingChain.gsm) using various methods can be found in the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

When modeling an aging chain consisting of specific age groups (e.g., 0 – 1 years, 1 – 2 years, etc.), it is often most appropriate to model the chain using a series of Material Delays.

Read more: Material Delay Elements (page 329).

Each Delay has a fixed delay time (with no dispersion) corresponding to the age cohort (e.g., 1 year). As such, the residence time in the cohort does not represent an average; it is an exact time required to “transit” the cohort. The structure of the model would typically look something like this:
Note that each Delay does not flow into the next Delay in the series. Rather, it flows through a Splitter, so that losses (e.g., due to deaths) can be accounted for prior to graduating to the next cohort. Of course, there could also be additional flows into and out of a cohort (due to immigration and emmigration). The Material Delay dialog for a typical cohort for this simple would look like this:

![Material Delay Properties dialog]

An example file which illustrates how aging chains can be modeled (AgingChain.gsm) using various methods can be found in the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).
In some types of aging chains, you would like to instantaneously “graduate” all the members of one cohort to the next cohort at a specific time (e.g., every year), and repeat the process for all cohorts. For example, if you had 6 age groups, at the beginning of each year, you might want to transition all the age groups to the next cohort (e.g., cohort 5 graduates to cohort 6, cohort 4 graduates to cohort 5, cohort 3 graduates to cohort 4, etc.).

To facilitate this, GoldSim provides specialized capabilities for Discrete Change and Integrator elements.

**Read more:** [Integrator Elements](page 228); [Discrete Change Elements](page 383).

One of the Instruction options for a Discrete Change transaction is a “Push”:

![Discrete Change Definition](image)

Push instructions can subsequently be processed by Integrators (by specifying the discrete change output as an input in the **Discrete Change** input field of the Integrator).

---

**Note:** Push discrete change outputs can only be processed by Integrator elements. Splitters and Allocators can also accept Push discrete changes, as they don’t actually process them; they simply pass them on (preserving the instruction). However, directing a Push discrete change to any other element (e.g., a Reservoir) will result in a fatal error.

---

When processing a Push discrete change signal, you must specify (under the Advanced options for an Integrator) that the Integrator itself, in turn, provides a "Pushed_Out" discrete change output:

![Advanced Options](image)

Doing so adds a new output called “Pushed_Out” to the element.

When an Integrator receives a Push discrete change signal (and the “Provide a Pushed_Out discrete change output” is checked), the following occurs:

1. The current value of the Integrator is instantaneously “pushed out” and replaced with value specified by the incoming discrete change signal.

2. The element produces a discrete change output named “Pushed_Out” that has a “Push” Instruction and a Value equal to the value of the Integrator prior to receiving the discrete change signal. This can subsequently be directed to another Integrator.
By creating a series of Integrators connected by Push discrete changes (triggered, for example, once per year), you can create an aging chain that instantaneously “graduates” all the members of one cohort to the next cohort at a specific time (e.g., every year). The structure (for a model with no other losses or additions) would look something like this:

The Integrator representing one of the cohorts would look like this:

![Integrator Properties](image)

**Note:** When a scalar Integrator receives a Push discrete change signal and the “Provide a Pushed_Out discrete change output” is NOT checked, the discrete change is treated as if it had an Add instruction (since it has nowhere to “push” the current value). Hence, the Integrator simply accumulates the changes.

An example file which illustrates how aging chains can be modeled (AgingChain.gsm) using a series of scalar Integrators can be found in the General Examples folder in your GoldSim directory (accessed by selecting **File | Open Example...** from the main menu).

Although this type of approach can represent specific age groups very accurately, it can be cumbersome if there are a large number of groups. An alternative to this approach that is much better at simulating a large number of age groups is to use an Integrator *array* that processes discrete pushes.
In this approach, instead of using a series of Integrators, you use a single Integrator that is a vector. A discrete change signal with a Push instruction is sent to the Integrator. When it receives the instruction, the Value associated with the discrete change is “pushed” into the first item of the vector. Each value in the vector is then pushed downward to the next item of the vector. That is, the value that was in item 1 is pushed to item 2; the value that was in item 2 is pushed to item 3, and so on.

If “Provide a Pushed_Out discrete change output” is checked for the Integrator, the element produces a discrete change output named “Pushed_Out” that has a “Push” Instruction and a Value equal to the value of the last item in the Integrator vector prior to receiving the discrete change signal. This can subsequently be directed to another Integrator. If “Provide a Pushed_Out discrete change output” is NOT checked for the Integrator, the final item in the Integrator vector accumulates all of the “pushes” that it receives.

When implementing such a vector aging chain, the Discrete Change element that generates the initial “push” into the Integrator must be defined in a specific manner. In particular, it must be defined as follows:

- The Instruction must be “Push”;
- The Value must be a scalar value; and
- The Type must be defined to match the Type of the Integrator that will receive the discrete change. Hence, although the Value itself is a scalar (since it represents the value being pushed into the first item of the vector), the Discrete Change element must be defined as a vector (as it will be an input to a vector Integrator).

An example of this is shown below:

This approach allows you to replace a series of Integrators with a single vector Integrator, and therefore facilitates simulation of chains with a large number of cohorts.

An example file which illustrates how aging chains can be modeled (AgingChain.gsm) using a vector Integrator can be found in the General
Solving Convolution Integrals

Examples folder in your GoldSim directory (accessed by selecting **File | Open Example...** from the main menu).

This approach can be extended to utilize a matrix Integrator. In this case, a discrete change signal (whose Value is defined as a vector) with a Push instruction is sent to the Integrator. When it receives the instruction, the Value associated with the Discrete Change is “pushed” into the first *row* of the matrix. Each row in the matrix is then pushed downward to the next row of the matrix. That is, the values that were in row 1 are pushed to row 2; the values that were in row 2 are pushed to row 3, and so on.

If “Provide a Pushed_Out discrete change output” is checked for the Integrator, the element produces a discrete change output named “Pushed_Out” that has a Push Instruction and a Value equal to the value of the last row in the Integrator matrix prior to receiving the Discrete Change signal. This can subsequently be directed to another Integrator. If “Provide a Pushed_Out discrete change output” is NOT checked for the Integrator, the final row in the Integrator matrix accumulates all of the “pushes” that it receives.

When implementing such a matrix aging chain, the Discrete Change element that generates the initial “push” into the Integrator must be defined in a specific manner. In particular, it must be defined as follows:

- The Instruction must be “Push”;
- The Value must be a vector having the same Array Label Set as the columns of the matrix into which it is being pushed; and
- The Type must be defined to match the Type of the Integrator that will receive the Discrete Change. Hence, although the Value itself is a vector (since it represents the values being pushed into the first row of the matrix), the Discrete Change element must be defined as a matrix (as it will be an input to a matrix Integrator).

**Solving Convolution Integrals**

Convolution elements are highly specialized elements in GoldSim that solve *convolution integrals*. Convolution integrals have many important applications in engineering, science and business.

The Convolution element is discussed in detail below.

Before describing how to use a Convolution element, it is first important to understand exactly what the element does mathematically, and what this mathematical operation represents conceptually.

Consider a dynamic system, in which an input signal, say \(x(t)\), enters a “black box”, \(S\), and is output as \(y(t)\):

\[
x(t) \rightarrow S \rightarrow y(t)
\]

Mathematically, we represent the “black box” system as \(S\):

\[
y(t) = S[x(0 \text{ to } t)]
\]

where \(y(t)\) represents the output signal at time \(t\), and \(x(0 \text{ to } t)\) represents the time history of the input signal. If the “black box” represents a convolution, it means that \(S\) takes on the following mathematical form:
In this equation \( h_{\tau}(t - \tau) \) is the transfer function (also referred to as the impulse response function). \( t - \tau \) is referred to as the lag, as it represents the time lag between the input and the output time. The transfer function is given a subscript \( \tau \) to indicate that the function itself may take on different forms at different points in time. That is, the response to an input impulse at \( t = \tau_1 \) could have a different form than the response to an impulse at \( t = \tau_2 \).

Convolution elements require that you supply the input signal, \( x(t) \), and the transfer function, \( h(t - \tau) \). The element then computes the output signal, \( y(t) \).

Note that the convolution integral is a linear operation. That is, for any two functions \( x_1(t) \) and \( x_2(t) \), and any constant \( a \), the following holds:

\[
S[x_1(t) + x_2(t)] = S[x_1(t)] + S[x_2(t)]
\]

\[
S[a x_1(t)] = a S[x_1(t)]
\]

Having defined mathematically what a convolution integral does, let us now try to understand what it represents conceptually. Perhaps the simplest way to think about the convolution integral is that it is simply a linear superposition of response functions, \( h(t - \tau_i) \), each of which is multiplied by the impulse \( x(\tau_i)\delta\tau \).

A more common way to interpret the convolution integral is that the output represents a weighted sum of the present and past input values. We can see this if we write the integral in terms of a sum (and assume here that the system is discretized by a single unit of time):

\[
y(t) = x(0)h(t) + x(1)h(t-1) + x(2)h(t-2) + \ldots
\]

The convolution operation may also be thought of as a filtering operation on the signal \( x(t) \), where the transfer function is acting as the filter. The shape of the transfer function determines which properties of the original signal \( x(t) \) are "filtered out."

Note that if the transfer function was a constant, the integral would collapse and the output signal would just be a scaled version of the input signal. Integration is necessary when the transfer function is not constant. Conceptually, this indicates that the system has memory and does not respond instantaneously to a signal. Rather, the response is delayed and spread out over time.

Using the Convolution Element

The property dialog for a Convolution element looks like this:
Convolution elements have a single output. You can specify these attributes by pressing the **Type...** button. By default, a new Convolution element is a scalar, dimensionless value. You can also use Convolution elements to operate on and/or create vectors and matrices.

**Read more:** Using Vectors and Matrices (page 782); Convolution Elements Defined as Arrays (page 821).

The **Display Units** field refers to the display units of the output signal (which may have different dimensions than the input signal).

You can save the results for a Convolution element by clicking **Final Values** and/or **Time Histories**.

Under **Signal Definition**, you first specify the **Input** signal. You must also specify the dimensions of the input signal (by defining **Units**). This facilitates error checking. The **Input** signal will be a direct or indirect function of time.

You then define the Transfer **Function**. In some cases, it may be easier to define the integral of the transfer function, rather than the transfer function itself, and the two radio buttons on the dialog allow you to choose between these two options.

The transfer function essentially “spreads out” or “disperses” a unit impulse over time. In many cases, the transfer function may be a continuous function with a long “tail”. That is, 99% of output produced by an impulse may be generated within 10 days of the impulse, with the remainder being spread out over the remaining 1000 days. Such a transfer function can have a significant computational impact on your model. Hence, GoldSim provides a way to “cut off” a transfer function after a certain lag time. If you check the **Truncate the Function at** checkbox, you can then enter a time after which the transfer function will be truncated. In the above example, you might, for example, specify this as 10 days. This input can be a function of time (e.g., if the transfer function changes with time, you may need to adjust the truncation time accordingly).
The Convolution element works by generating and continuously adding to a projected future history of its output. At each timestep the incremental input is convolved with the transfer function and added to that output history. The history stores projected results for every scheduled timestep of the model, and also inserts additional points where necessary to ensure that the difference between successive time points is never greater than 10% of the **Truncate the Function at** value.

However, if your model generates unscheduled updates and if the duration of the transfer function is short, it is possible that the projected future history will not have enough time points to deliver accurate results. In this situation you can use the optional input **Maximum prediction timestep** to ensure sufficient accuracy of this history. The must be specified as a time, and cannot have any links. It should be short enough so that it can reasonably approximate the shape of the transfer function.

Note that in the definition of a convolution integral, the transfer function is mathematically defined not as a function of time, but as a function of a time difference, \( (t - \tau) \), or lag time. In order to write such an equation in GoldSim, you reference the time difference \( t - \tau \) using a special locally available property of the Convolution element called “Lag”. It must be referenced as “~Lag”.

Note: As indicated by the way it is referenced, the Lag is an example of a locally available property. As such, it only has meaning inside the Transfer Function field, and cannot be referenced anywhere else.

**Read more:** [Understanding Locally Available Properties](#) (page 808).

For example, if your transfer function was the following equation:

\[
h(t - \tau) = Ce^{-a(t - \tau)/b}
\]

then you would reference the lag time \( (t - \tau) \) as “~Lag”:

The transfer function does not have to be time-invariant. That is, the function can be defined to change as a function of time. Referring to the example above, if the response was different at later times in the simulation (e.g., after 1 yr), the transfer function could be written as follows:

GoldSim checks to ensure that the dimensions of the various inputs and outputs are related as follows:

\[
\text{Output} = \text{Transfer Function} \times \text{Input Signal} \times \text{time}
\]

If the integral of the transfer function is specified, GoldSim checks to ensure that the dimensions of the various inputs and outputs are related as follows:
Output = Integral of Transfer Function * Input Signal

For example, if the element display units were specified as $ and the input signal display units were specified as g/yr, then GoldSim would require that the transfer function had dimensions of currency/mass (e.g., $/g).

Several points should be noted regarding the numerical implementation of the Convolution element within GoldSim:

- During the numerical integration, GoldSim does not actually start the integration from 0. Instead, it starts the integration from a very small number. This allows the transfer function to be a function of the inverse of the lag without causing numerical problems (i.e., divide by zero errors)
- The numerical integration approach used by GoldSim assumes that the input function varies linearly between time steps; and
- To have acceptable accuracy, the simulation timestep should be short relative to the rate of change of the transfer function. Note, however, that your model may run much slower if the timestep is too small, as the computational effort for solving a Convolution element goes up as the square of the number of timesteps.

Convolution Elements Defined as Arrays

A convolution element can be defined to output an array. In this case, the dialog appears like this:

In this case, the Input Signal must be specified as an array. Pressing the Edit Vector… button (or Edit Matrix… button if it is a matrix) provides access to a dialog for defining the scalar transfer function for each separate item of the array:

Examples of the Use of the Convolution Element

In order to fully understand how to use Convolution elements, it is helpful to explore several examples. The three examples discussed here can be found in the General Examples folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu), in a file named Convolution.gsm.

We examine three simple examples below:

- A simple filter;
- Computing costs associated with warranties; and
- Computing response of a structure due to an earthquake.

Convolution Example: A Simple Filter

This example shows how a simple input signal can be filtered to alter its shape. In this example, the input signal is a square “boxcar” function:
Mathematically, this can be expressed in GoldSim as follows:

\[
\text{If}(\text{ETime} < 5 \text{ days OR ETime} > 10 \text{ days}, 0, 1)
\]

The Transfer function is also a square “boxcar” function, in which any impulse is delayed for 10 days, and then spread out evenly over 5 days:

Mathematically, this can be expressed in GoldSim as follows:

\[
\text{If}(\neg \text{Lag} < 10 \text{ days OR } \neg \text{Lag} > 15 \text{ days}, 0, 0.2)
\]

This input signal and transfer function produce the following output signal:
Solving Convolution Integrals

**Convolution Example: Warranty Costs**

Note that the output signal in this case has dimensions of time (since the input signal and the transfer function were defined as being dimensionless).

This example shows how warranty costs for a product (e.g., a vehicle) can be computed based on a product sales rate and the expected warranty cost time history for a product using a Convolution element.

In this example, the input signal is the sales rate of the vehicle:

Sales will start one year in the future, and ramp slowly up for three years. The vehicle is only assumed to be produced for three years.

This can be entered into GoldSim as a Time Series.

The Transfer function represents, on average, how warranty costs are expected to be incurred in the future (per vehicle):
This curve indicates that warranty costs start to be incurred about 1.5 years after the sale, reach a peak at 3 years after the sale, and no more costs can be expected to occur after about 4.5 years after the sale. Note that the units on the transfer function are $/car/yr (the curve integrates to 2000 $/car).

The output signal will then have dimensions of $/yr:

\[
\text{Output} = \text{Transfer Function} \times \text{Input Signal} \times \text{time}
\]

\[
\text{Output} = \$/\text{car/yr} \times \text{cars/yr} \times \text{yr} = \$/\text{yr}
\]

That is, the output represents the rate at which we can expect to incur warranty costs for all of the vehicles sold:

If we integrate the warranty cost rate (using an Integrator), we can compute the cumulative warranty costs:
This example shows how a Convolution element can be used to simulate how a structure responds to an earthquake. The input signal is an accelerogram (a time history of ground accelerations). The output signal is the displacement of a building (illustrating how the building oscillates from side to side in response to the earthquake).

The input signal (the accelerogram) is taken from an actual earthquake in Southern California in 1940 (the El Centro earthquake):

As can be seen, the earthquake lasted about 30 seconds.

For simplicity, the building is treated as a damped spring. The transfer function under such an assumption is represented by the following equation:

\[-\frac{e^{-n(t-r)}}{W_d} \sin[W_d(t - r)]\]

where \( n \) is the coefficient of decay (in radians/sec), and \( W_d \) is the damped frequency (in radians/sec). A plot of this transfer function looks like this:
Generating Stochastic Time Histories

The transfer function represents the ratio of a displacement response to a “pulse” of ground velocity, and has dimensions of time.

This curve indicates that once excited by an impulse, the building oscillates with the oscillations dying out within about 15 seconds.

The output signal will have dimensions of length:

\[
\text{Output} = \text{Transfer Function} \times \text{Input Signal} \times \text{time}
\]

\[
\text{Output} = \text{sec} \times \text{m/\text{sec}^2} \times \text{sec} = \text{m}
\]

That is, the output represents the displacement of the building in response to the earthquake:

---

**Generating Stochastic Time Histories**

It is sometimes necessary to generate stochastic time histories of variables. A stochastic time history is a random time history that is generated according to a specified set of statistics. For example, for one type of stochastic time history, the "random walk", the variable can be thought of as taking successive steps,
each in a random direction. The overall trend (up or down) and the size of the steps are controlled by the statistical measures specified by the user.

GoldSim provides a specialized element to generate such time histories, the History Generator. This element is primarily envisioned as a tool to allow users to simulate financial and economic variables (e.g., security prices, interest rates), but has many applications in other arenas.

The default icon for the History Generator element looks like this:

![History Generator Icon]

The property dialog looks like this:

![Property Dialog]

Like all GoldSim elements, you first specify an **Element ID** and a **Description**. Below the **Description** field, you specify the **Display Units** you wish to use.

To the right of the **Display Units**, you can specify the **Type**. By default, a History Generator provides a scalar output (and requires scalar inputs). However, via this button, you can also specify it to have vector (1-D array) output (and require vector inputs).

The "History Definition" portion of the History Generator dialog is used to specify the characteristics of the stochastic history. The **History Type** field is used to select the general type of stochastic history. GoldSim currently provides two options: "Geometric Growth" and "Random Walk". The details of how the History Definition fields are specified are provided below.

An example file which illustrates the use of History Generator elements (HistoryGenerator.gsm) can be found in the General Examples folder in your GoldSim directory (accessed by selecting **File | Open Example...** from the main menu).
Before creating stochastic time histories using the History Generator element, it is instructive to briefly summarize the different kinds of time histories that can be generated.

Consider the stochastic time histories below:

These are random histories that follow no particular trend. They simply randomly walk through time. Clearly one of the curves has larger random fluctuations than the other. For certain types of variables (e.g., stock prices), the variability of the time history as a function of time is described in terms of a volatility. The history with greater fluctuations is said to be more "volatile".

Note that the volatility is often a function of the unit of time on which it is based. In the examples shown above, the volatility increases with time (i.e., the annual volatility is greater than the daily volatility). For other types of stochastic variables, however, the volatility may stay constant with time. In the example below, the variable represented by the lower curve is target-reverting. That is, the value reverts to some underlying target (in this case, a constant value of 1). As a result, the volatility is constant in time.

The histories in the two examples above either move randomly through time, or move randomly around some constant target. Other stochastic histories may follow a trend.
The histories below grow geometrically at a rate of 5% per year (this is referred to as geometric Brownian motion):

The dark smooth line is growth with no volatility. The other two lines both are volatile. In one, the volatility grows with time. In the other, the volatility stays constant.

If we were to run multiple realizations of the time history whose volatility grows with time, the growth in the volatility as a function of time would be more noticeable:

Compare this to multiple realizations of the time history whose volatility stays constant with time:
Of course, the underlying trend does not need to be growth (positive or negative). The histories below follow a cyclical trend:

The dark smooth line is the signal with no volatility. The other two lines are both volatile. In one, the history lags the underlying trend. In the other, there is no lag.

The History Generator element can be used to generate all of these types of stochastic time histories.

Generating Geometric Growth Histories

Selecting "Geometric Growth" for the History Type field of a History Generator element allows you to define a history that grows geometrically with time. This means that the rate of growth and the amount of volatility are proportional to the current value. Among other things, this type of history is appropriate for variables such as security prices.
In its simplest form, this type of history requires three inputs (Mean Annual Growth Rate, Annual Volatility, and Initial Value). The other two inputs (Annual Reversion Rate and Initial Value of Median) are only required if you wish the history to revert toward its median (and this is discussed in the next section).

**Mean Annual Growth Rate.** This is the mean annual logarithmic growth rate ($\mu'$), where $V_i$ is the value at time $i$:

$$
\mu' = \frac{1}{n} \sum_{i=i-n+1}^{i} \ln \left( \frac{V_{i+1}}{V_i} \right)
$$

Although this is a rate (i.e., time-based), because it is "hard-wired" to be an annual value, this input is dimensionless. That is, it represents the logarithmic growth rate over a year. It can be positive, negative, or zero.

If the logarithmic growth rate is measured over a different time period (e.g., monthly), this can be scaled by direct multiplication. For example, a mean monthly logarithmic growth rate can be converted to a mean annual logarithmic growth rate by multiplying it by 12.

**Annual Volatility.** This is the standard deviation of the annual logarithmic growth rate ($\sigma'$):

$$
\sigma' = \sqrt{\frac{1}{n-1} \sum_{i=i-n+1}^{i} \left( \ln \left( \frac{V_{i+1}}{V_i} \right) - \mu' \right)^2}
$$

This is a dimensionless value that must be non-negative.

**Note:** As pointed out above, the volatility for a geometric growth history is, by definition, dimensionless. This is different than the volatility for a random walk history, which has dimensions.

If the logarithmic volatility is measured over a different time period (e.g., monthly), and the rate of reversion to the median is zero, this can be scaled by the square root of the time period. For example, a mean monthly logarithmic volatility can be converted to a mean annual logarithmic volatility by multiplying it by $\sqrt{12}$. 

- GoldSim User's Guide
- Chapter 10: Advanced Modeling Concepts • 831
Initial Value. This is the initial value of the time history (the value at time zero). It has the same dimensions as the output. It must be a positive number.

If the Annual Reversion Rate is set to zero (the default), GoldSim generates successive values as follows:

\[ V_{\text{new}} = V_{\text{old}} \exp \left( \mu' \Delta t + \sigma' \sqrt{\Delta t} \right) \]

where \( V_{\text{new}} \) is the new value, \( V_{\text{old}} \) is the previous value, \( \Delta t \) is the time between the two values, \( \varepsilon \) is a random standard normal value (sampled from a distribution with mean 0 and standard deviation 1), and \( \mu' \) and \( \sigma' \) are as defined above.

For constant values of the Mean Annual Growth Rate and Mean Annual Volatility, this results in a history that (on average) grows exponentially, and whose standard deviation of the logs increases as the square root of time:

At any given time, the distribution of values is log-normal.

The logarithmic input definitions for the Growth Rate and Volatility, as used by GoldSim, are those typically used by academic financial analysts. However, it may be difficult to find data expressed using these formal definitions. Some simple approaches for converting available data to these forms are as follows:

Converting from Geometric Mean Returns. Typically, the performance of individual securities is reported as "average annual total returns". Mathematically, this actually represents the geometric mean of the annual returns over a period:

\[ \mu_g = \left( \prod_{i=1}^{n+1} \left( \frac{V_{i+1}}{V_i} - 1 \right) \right)^{1/n} - 1 \]

You can use GoldSim's built in gm2cm function to convert a geometric mean annual return to an arithmetic mean log annual return.

If the standard deviation of the average annual total returns is also available:
then this can be used (in conjunction with the geometric mean of the returns) to compute the volatility using GoldSim's geo2vol function.

**Read more:** [Financial Functions](page 128).

**Converting from Arithmetic Mean Returns.** In some cases, you may have return rate statistics reported in terms of the arithmetic mean of the annual return over a period ($\mu_a$, defined above).

You can use GoldSim's built in ari2cm function to convert an arithmetic mean annual return (and a standard deviation) to an arithmetic mean log annual return, and the ari2vol function to convert a standard deviation (and an arithmetic mean) of the annual return to the standard deviation of the log annual return (the volatility).

By default, the standard deviation of a geometric growth type history increases with the square root of time. In some cases, when simulating a stochastic history that increases geometrically, you may not want the standard deviation to grow in this way. Rather, you may want it to stabilize at a constant value.

You can accomplish this by forcing the stochastic history to revert towards its median value at a specified rate. To do so, you must use the **Annual Reversion Rate** and **Initial Value of Median** input fields.

**Annual Reversion Rate.** This is the annual fractional rate at which the history reverts towards its median. Although this is a rate, because it is "hard-wired" to an annual value, this input is dimensionless. That is, it represents the fractional rate per year. It must be non-negative.

**Initial Value of Median.** In most cases, the initial value of the time history is unlikely to be at the median. Hence, this is the initial value of the median of the time history (the value at time zero). It has the same dimensions as the output. It must be a positive number.

If the **Annual Reversion Rate** is non-zero, GoldSim generates successive values as follows:
Generating Stochastic Time Histories

\[ V_{\text{new}} = V_{\text{old}} \exp \left( \mu' \Delta t + \left( 1 - e^{-r \Delta t} \right) \ln T_{\text{old}} - \ln V_{\text{old}} \right) + \sigma' \sqrt{\frac{1 - e^{-2r \Delta t}}{1 - e^{-2r(1\text{yr})}}} \]

where \( V_{\text{new}} \) is the new value, \( V_{\text{old}} \) is the previous value, \( \Delta t \) is the time (in years) between the two values, \( \varepsilon \) is a random standard normal value (sampled from a distribution with mean 0 and standard deviation 1), \( r \) is the reversion rate, and \( \mu' \) and \( \sigma' \) are the logarithmic growth rate and volatility, respectively. \( T \) is the median (a function that grows geometrically from the specified Initial Value of Median using the mean annual growth rate). \( T_{\text{old}} \) is the previous value of the median.

In practical terms, specifying a non-zero Annual Reversion Rate has two effects:

- The history tracks the median, and if the Initial Value differs from the Initial Value of Median, the history approaches the median history with a half-life of \( \ln(2)/\text{Annual Reversion Rate} \).
- The standard deviation of the logs initially grows, approaching a constant value with a similar half-life. The "steady-state" standard deviation of the log of the values stabilizes at a value of:

\[ \sigma' \sqrt{\frac{1}{1 - e^{-2r}}} \]

Generating Random Walk Histories

Selecting "Random Walk" for the History Type field of a History Generator element allows you to define a history that randomly walks through time.

In its simplest form, this type of history requires two inputs (Annual Volatility, and Initial Value). The other two inputs (Target and Annual Reversion Rate) are only required if you wish the history to track an underlying target or trend (and this is discussed in the next section).

**Annual Volatility.** This is the standard deviation of the annual values (\( \sigma \)):

\[ \sigma = \sqrt{\frac{\sum_{i=t-n}^{t} [V_i - \mu]^2}{n-1}} \]

where \( \mu \) is the mean of the annual values:

\[ \mu = \frac{\sum_{i=t-n}^{t} V_i}{n} \]

If the volatility is measured over a different time period (e.g., monthly), this can be scaled by direct multiplication. For example a mean monthly volatility can be converted to a mean annual volatility by multiplying it by \( \sqrt{12} \).

It has the same dimensions as the output, and must be non-negative.
Initial Value. This is the initial value of the time history (the value at time zero). It has the same dimensions as the output and can be positive, negative or zero.

You can control whether your history can take on negative values using the Allow Negative Values checkbox (which defaults on). If this box is cleared, negative values are “truncated”. Note that this has the effect of artificially reducing the standard deviation of the values.

If the Annual Reversion Rate is set to zero (the default), GoldSim generates successive values as follows:

\[ V_{\text{new}} = V_{\text{old}} + \sigma \varepsilon \sqrt{\Delta t} \]

where \( V_{\text{new}} \) is the new value, \( V_{\text{old}} \) is the previous value, \( \Delta t \) is the time (in years) between the two values, \( \varepsilon \) is a random standard normal value (sampled from a distribution with mean 0 and standard deviation 1), and \( \sigma \) is the volatility.

For constant values of the Annual Volatility, this results in a history whose standard deviation increases as the square root of time.

By default, the standard deviation of a random walk type history increases with the square root of time. In some cases, when simulating such a history, you may not want the standard deviation to grow with time. Rather, you may want it to stabilize to a constant value around an underlying target.

You can accomplish this by forcing the stochastic history to revert towards a target value at a specified rate. To do so, you must use the Annual Reversion Rate and Target input fields.
Target. This is the underlying target that the history will revert to (i.e., track). It can be a constant value or a function of time. It has the same dimensions as the output. It can take on any value.

Annual Reversion Rate. This is the annual fractional rate at which the history reverts to its target. Although this is a rate, because it is "hard-wired" to an annual value, this input is dimensionless. That is, it represents the fractional rate per year. It must be non-negative.

If the Annual Reversion Rate is non-zero, GoldSim generates successive values as follows:

\[ V_{\text{new}} = V_{\text{old}} + \left(1 - e^{-r \Delta t}\right) \left(T_{\text{new}} - V_{\text{old}}\right) + \sigma \sqrt{\frac{\ln(2) \Delta t}{1 - e^{-2r(1\text{yr})}}} \]

where \( V_{\text{new}} \) is the new value, \( V_{\text{old}} \) is the previous value, \( \Delta t \) is the time (in years) between the two values, \( \varepsilon \) is a random standard normal value (sampled from a distribution with mean 0 and standard deviation 1), \( r \) is the reversion rate, and \( \sigma \) is the volatility. \( T_{\text{new}} \) is the current (new) value of the underlying target (which may change with time).

In practical terms, specifying a non-zero Annual Reversion Rate has two effects:

- The history tracks the target, and if the Initial Value differs from the initial value of the Target, the history approaches the target history with a half-life of \( \ln(2)/\text{Annual Reversion Rate} \).
- The standard deviation initially grows, but eventually stabilizes to a constant value with a similar half-life. The "steady-state" standard deviation stabilizes at a value of:

\[ \sigma \sqrt{\frac{1}{1 - e^{-2r}}} \]

By default, when an Annual Reversion Rate is specified, the history actually lags the target, and as a result, the signal is "dispersed" such that the peaks and valleys of the history are (on average) smaller than that of the target:

The time lag increases as the Annual Reversion Rate decreases.
In some cases, you may not want such a time lag. The **Do not lag Target Changes** checkbox (if checked) eliminates this lag. In this case, GoldSim generates successive values as follows:

\[
V_{\text{new}} = V_{\text{old}} + \left(1 - e^{-r \Delta t}\right) \left(T_{\text{old}} - V_{\text{old}}\right) + \sigma \sqrt{\frac{1 - e^{-2 \pi t \Delta t}}{1 - e^{-2 \pi (1 \text{yr})}} \left(T_{\text{new}} - T_{\text{old}}\right)}
\]

As can be seen below, this has the impact of eliminating the lag (and the dispersion):

---

**Simulating Correlated Arrays of Stochastic Histories**

A History Generator can be specified to represent a 1-D array of variables (i.e., a vector). In this case, the various input parameters must be specified as vectors.

This most common application of this is to simulate an array of correlated variables (e.g., a portfolio of investments). When variables are correlated, their random movements (as typically quantified by the volatility) are linked to a greater or lesser degree. For example, consider the three time series below:

In this example, the volatile movements of Beta and Gamma are strongly correlated, and are both uncorrelated to Alpha.
The correlations between the members of an array are specified via a correlation matrix. A correlation matrix specifies the correlations between variables, and generally has the following form:

<table>
<thead>
<tr>
<th></th>
<th>Alpha</th>
<th>Beta</th>
<th>Gamma</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha</td>
<td>1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Beta</td>
<td>0.1</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>Gamma</td>
<td>0.1</td>
<td>0.99</td>
<td>1</td>
</tr>
</tbody>
</table>

Note that by definition, a correlation matrix is symmetric around its diagonal (since the cross diagonal terms define the same correlation coefficient).

If you specify that the History Generator represents a 1-D array of variables (i.e., a vector), then the **Use correlation matrix** checkbox becomes available on the History Generator dialog:

If this box is checked, the **Edit Matrix**... button becomes available. This button provides access to a dialog for specifying the correlation matrix:

By default, all off-diagonal correlation coefficients are zero. The matrix is symmetrical, so you need only define one of the cross-diagonal terms. The value represents a rank correlation coefficient, and must vary between -1 and 1. It must be a number (i.e., you cannot specify a link).

**Note:** When you define a correlation matrix, it is important to ensure that it is internally consistent. For example, if you specified that A was positively correlated to B, and B was positively correlated to C, but that A was negatively correlated to C, the correlation matrix would be inconsistent (since in this case, A should also be positively correlated to C). When this occurs, GoldSim will produce a fatal error message.
GoldSim provides several different algorithms for correlating the members of the vector. These are selected from the Correlation Algorithm drop-list at the top of the dialog. The various correlation algorithms are discussed in detail in Appendix B.

Using Resources

One of the advanced features in GoldSim is the ability to create and interact with Resources. A Resource is something that has a limited supply (e.g., spare parts, fuel, skilled personnel, money) and is required in order for elements of the modeled system to carry out certain actions.

For example, in order for a manufacturing process to proceed, perhaps two different parts and a skilled operator must be available. This could be modeled by defining an Event Delay (to represent the process) with three Resource requirements that must be met in order for the element to process the event.

Alternatively, in order for a project task to continue to operate, perhaps a certain amount of cash must be available. This could be modeled by defining a Conditional Container (to represent the task) with a single Resource requirement (a spend rate) that must be met in order for the task to continue (i.e., for the Container to stay active).

Only certain elements in GoldSim can interact with Resources (two of these are mentioned in the examples above: an Event Delay and a Conditional Container). Elements can consume, borrow, and in some cases, generate Resources.

Using Resources involves three different steps:

- You must define a Resource Type (e.g., Gasoline, Electricians, Pump Motors) and specify its characteristics (discrete or continuous).
- You must specify where the Stores for that particular Resource exist in your model. Stores represent stockpiles or places where the actual Resource (e.g., parts, personnel) is stored or located when not being used. That is, Resource Stores can be thought of as having physical locations in the system you are modeling.
- You must specify how particular elements in the model interact with (consume, borrow from, or add to) the Resource Stores.

The sections below discuss the use of Resources in detail.

A simple example which illustrates the use of Resources (Resources.gsm) can be found in the General Examples folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

A Resource is something that has a limited supply (e.g., spare parts, fuel, skilled personnel, money) and is required in order for elements of the modeled system to carry out certain actions.

Creating a Resource requires two distinct steps:

1. First you must define the type of Resource (e.g., Gasoline, Electricians, Pumps, Motors) and specify its characteristics (discrete or continuous).
2. Then you must specify where the Stores for that particular Resource exist in your model.

Stores represent stockpiles or places where the actual Resource (e.g., parts, personnel) is stored or located when not being used. That is, Resource Stores can be thought of as having physical locations in the system you are modeling.
There are two types of Stores that can be created: **Global Stores** and **Local Stores**. Global Stores are available to all elements in your model. Local Stores are associated with specific Containers, and are only available to elements inside the Container (or in the case of a Conditional Container, to the Container itself).

A Resource Type can have multiple Stores in a model. For example, you could define multiple Local Stores, or perhaps a Global Store and several Local Stores. These two steps are described in the sections below.

You define or edit Resource types by using the Resource Manager, which is accessed via the main menu (Model | Resources…) or by pressing F8. The following dialog will be displayed:

Pressing the Add… button displays the following dialog for defining a new Resource Type:
Each Resource Type requires a unique **Resource ID**. Resource IDs follow the same rules as elements names (e.g., no spaces, letters, only numbers and the underscore character can be used; cannot start with a number). You should also provide a brief **Description**.

A Resource has the following attributes:

- **Type**: This can be “Continuous” or “Items”. Continuous should be used for things that are not discrete (e.g., fuel). Items should be specified for things that are discrete (e.g., parts, personnel).

- **Units**: If the Resource is specified as Continuous, in most cases you will want to specify units (e.g., m$^3$, kg). If the Resource is specified as Items, units are not applicable.

- **Order**: A Resource can be defined as a scalar or a vector. Defining a Resource as a vector is most useful, for example, if you have a large number of individual parts. Defining these as a vector is much easier than defining a separate Resource for each part.

  **Read more**: Using Vectors and Matrices (page 782).

- **Labels**: If you define the Resource as a vector, you need to specify a set of Array Labels from the Labels drop-list.

  **Read more**: Understanding Array Labels (page 783).

**Note**: The bottom part of the dialog provides an option to create a “Global Store” of this particular Resource. This is discussed in detail in the next section.

If multiple Resource Types are defined, you can move between them using the **Previous** and **Next** buttons.

After you have finished defining the Resource, press the **OK** button to return to the Resource Manager dialog:
You can delete and edit existing Resource Types using the **Delete** and **Edit** buttons, respectively.

**Note:** If a selected Resource Type is being used by an element in the model, or has a Local Store defined, GoldSim will display an error message if you try to delete it. You can only delete Resource Types that are not being used.

The **Up** and **Down** buttons move Resources up and down the list (this is simply cosmetic; the order in which they are listed has no impact).

Note that you can also access the Resource Manager by right-clicking in the graphics pane at the top level of the model and selecting Properties. The Resource Manager is available as a tab on the Model Properties dialog:

**Creating and Editing Global Resource Stores**

Global Resource Stores are available to all elements in your model. Global Stores are created within the Resource Manager (accessed via **Model** | **Resources**… or by pressing **F8**). When you **Add** or **Edit** a Resource Type, you have the option to define a Global Store for that Resource by checking **Define a Global Store for this Resource**.
This creates a Global Store and provides access to the three input fields to define the Store: the Initial Quantity, the Rate of Addition and Discrete Additions.

The Initial Quantity must have the same attributes (order and dimensions) as the Resource Type.

**Note:** The Initial Quantity must be a number or a link from a static variable (e.g., a constant Data element or a Stochastic).

The Rate of Addition allows you to add to your Resource Store in a continuous manner. It is not provided if the Resource Type is “Items”. It must have the same order as the Resource Type, but the dimensions must represent a rate of change (e.g., if the Resource Type has units of mass, the Rate of Addition must have units of mass per time). The Rate of Addition must be entered as a non-negative value. A negative value for this input during a simulation will result in a fatal error.

**Note:** The specified Rate of Addition represents a constant rate over the next timestep. Hence, if the Rate of Addition was defined as “if(time > 10 day, 2 m³/day, 1 m³/day)”, and you were using a 1 day timestep, the rate would not actually change to 2 m³/day until time = 11 days. That is, since at time = 10 days, the if statement indicates that the rate is equal to 1 m³/day, GoldSim would assume that the rate was equal to 1 m³/day between 10 days and 11 days. If you wanted the rate to change at 10 days, you would write the if statement as “if(time >= 10 day, 2 m³/day, 1 m³/day)”.

In addition to a continuous Rate of Addition, Resource Stores can also accept discrete changes, so that your Resource Store can change in a discrete manner. The Discrete Additions field only accepts discrete change signals (e.g., the output of a Discrete Change element). The discrete change signal must have a non-negative value. However, it can have either an Add instruction or a Replace instruction.

The Users button summarizes the properties of the Store, and lists all the elements that directly use this Store.
After you have created a Global Store, you will notice that Resource Types that have Global Stores are marked in bold in the Resource Manager:

The **Locations** button summarizes the properties of all of the Resource Stores in the model, as well as all of the elements that directly use those Stores.

If **Save Histories** is checked, GoldSim saves time histories of all of the Resource Stores in the model and users of those Stores. After the model is run, these histories can be viewed via the **Histories** button.

**Creating and Editing Local Resource Stores**

Local Resource Stores are associated with specific Containers, and are only available to elements inside the Container (or in the case of a Conditional Container, to the Container itself).

Local Stores are created via the Container dialog. Selecting the **Provide Resources** checkbox adds a **Resources** tab to the Container that allows you to create Local Stores:
Warning: When you provide Resources to a Container, the Treat as SubSystem feature is also automatically selected (and cannot be deselected unless you first turn off Provide Resources). That is, a Container with a Resource Store, by definition, is treated as a SubSystem, and this puts certain limitations on how these Containers can be used.

Read more: Treating a Container as a SubSystem (page 135).

The Resources tab for a Container looks like this:
Selecting the **Resources** tab and pressing the **Add** button displays the following dialog for selecting the Resource Type for which you want to create a Local Store:

![Select from available Resource Types dialog](image)

All *available* Resource Types will be listed. Note that a defined Resource Type is available as long as a Store for that Type has not already been created in the Container. That is, there can only exist a single Store of each Type in any given Container.

**Note:** If there is only one Resource Type in the model and it is available for use in the Container, the dialog above will be skipped and the single Resource Type will be automatically selected.

After selecting the Resource Type, the following dialog will be displayed:

![Edit Resource Store dialog](image)

This creates a Local Store and provides access to the three input fields to define the Store: the **Initial Quantity**, the **Rate of Addition** and **Discrete Additions**.

The Initial Quantity must have the same attributes (order and dimensions) as the Resource Type.

**Note:** The Initial Quantity must be a number or a link from a static variable (e.g., a constant Data element or a Stochastic).

The Rate of Addition allows you to add to your Resource Store in a continuous manner. It is not provided if the Resource Type is “Items”. It must have the same order as the Resource Type, but the dimensions must represent a rate of change (e.g., if the Resource Type has units of mass, the Rate of Addition must have units of mass per time). The Rate of Addition must be entered as a non-
negative value. A negative value for this input during a simulation will result in a fatal error.

**Note:** The specified Rate of Addition represents a constant rate over the next timestep. Hence, if the Rate of Addition was defined as “if(time > 10 day, 2 m³/day, 1 m³/day)”, and you were using a 1 day timestep, the rate would not actually change to 2 m³/day until time = 11 days. That is, since at time = 10 days, the if statement indicates that the rate is equal to 1 m³/day, GoldSim would assume that the rate was equal to 1 m³/day between 10 days and 11 days. If you wanted the rate to change at 10 days, you would write the if statement as “if(time >= 10 day, 2 m³/day, 1 m³/day)”.

In addition to a continuous Rate of Addition, Resource Stores can also accept discrete changes, so that your Resource Store can change in a discrete manner. The Discrete Additions field only accepts discrete change signals (e.g., the output of a Discrete Change element). The discrete change signal must have a non-negative value. However, it can have either an Add instruction or a Replace instruction.

The Users button summarizes the properties of the Store, and lists all the elements that directly use this Store.

Read more: Summarizing Resource Locations and Users (page 855).

After you have added one or more Local Stores, the Resource tab for a Container shows all of the Resource Stores associated with the Container:

You can delete and edit existing Resource Stores using the Delete and Edit buttons, respectively.

**Note:** If a Resource Store is being used and you delete it, GoldSim will allow this, but GoldSim will display an error when you try to run the model.
The **Up** and **Down** buttons move Resources up and down the list (this is simply cosmetic; the order in which they are listed has no impact).

The **Locations** button summarizes the properties of all Stores that exist in the Container, as well as all of the elements that directly use those Stores.

*Read more:* [Summarizing Resource Locations and Users](#) (page 855).

If **Save Histories** is checked, GoldSim saves time histories of all of the Resource Stores in the model and users of those Stores. After the model is run, these histories can be viewed via the **Histories** button.

*Read more:* [Viewing Resource Results](#) (page 859).

Once you have defined some Resources and created one or more Resource Stores, you can specify ways in which elements in your model interact with those Resource Stores.

Elements can interact with a Resource Store in three different ways:

- An element can *Spend* (consume) a Resource from the Store.
- An element can *Borrow* a Resource from the Store for a particular amount of time and then return it.
- An element can *Deposit* (generate) a Resource into a Store.

These interactions can be discrete or continuous in nature. As a result, there are actually five different kinds of possible Resource interactions:

- You can Spend a discrete amount of a Resource;
- You can Borrow a discrete amount of a Resource;
- You can Deposit a discrete amount of a Resource;
- You can Spend a Resource at a specified rate; and
- You can Deposit a Resource at a specified rate.

Only certain kinds of elements in GoldSim can interact with Resources, and not all five types of interactions may be available to a particular element.

Whenever an element does interact with a Resource Store, it does so through a **Resources...** button:

![Resources... button](image)

Depending on the context, this button is either available on the main dialog for the element or within a Trigger dialog for the element. When you press the button, the following dialog is displayed:

![Discrete Change Delay dialog](image)
A Resource Requirement (or more generally, an interaction) is created by pressing the Add button. When you do this, GoldSim will add a row to the table of Resource Requirements:

You must then edit the row to define the Resource Requirement. The Resource column is a drop-list that contains all Resource Stores that are available to the element.

What Stores are available to an element is a function of where the element is located in the model and the defined hierarchy of Local and Global Stores. The following rules are applied:

- An element can only interact with Resource Stores that are above it or at the same level within its Container path. For example, if the element is located within Container1/Container2 (i.e., Container2 which is inside Container1), the element can only interact with Local Stores associated with Container1 or Container2, and Global Stores. The element would not be able to interact with any Local Stores associated with Container3 (a Container that was at a parallel level to Container1).

- If a Resource Type has multiple Stores within a particular Container path, the element can only access the Store that is closest to it. For example, if the element is located within Container1/Container2 (i.e., Container2 which is inside Container1), and Container1 and Container2 have Local Stores of a particular Resource Type, and there is also a Global Store of that Type, the element can only access the Store in Container2, even if that Store becomes empty during the simulation. If the Store in Container2 was deleted, it could then only access the Store in Container1. Only if the Store in Container1 was also deleted could it access the Global Store.

In some cases, if one Resource Store is exhausted, you may want to move Resources from another Store. This can be done, but you must manually program how and when this takes place.

Read more: Moving Resources in a Model (page 853).

The Item column is not editable if the Resource is a scalar. However, if it is a vector, the column provides a drop-list for you to select the item from the vector for which the Resource Requirement is defined:
The **Usage** column specifies the type of interaction (e.g., Spend, Borrow, etc.). As noted above, there are five possible types of interactions, but depending on the element type and the Resource Type, not all of these will be available (e.g., a Spend Rate is not available for Resource Types that are discrete items).

The **Quantity/Rate** column is an input field in which you specify a value that is appropriate for the particular type of Usage (e.g., discrete usages require quantities, continuous usages require rates). You can enter a value or an expression (or links to other elements). It can also be a function of time. However, the dimensions must be consistent with the specified Resource Type. Within this field, you can also reference several locally available properties pertaining to the amount of Resource that is available, or has been spent or borrowed.

**Read more:** [Referencing Resource Availability and Use in Input Expressions](page 852).

**Note:** If a Resource becomes exhausted (due to Spend Rate) between scheduled updates (timesteps), GoldSim will insert an unscheduled update to accurately capture this. If you have disabled unscheduled updates (an advanced timestepping option), such that GoldSim cannot insert an unscheduled update when the Resource becomes exhausted, a fatal error will be displayed.

**Read more:** [Understanding Timestepping in GoldSim](page 455).

A Resource Requirement can be removed by pressing the **Delete** button. The **Up** and **Down** buttons move Resource Requirements up and down the list (this is simply cosmetic; the order in which they are listed has no impact).

**Note:** If you have multiple Resource Requirements, they are only applied when they can all be met. For example, if you had a Spend requirement of 3 of PartB and 2 of PartC, and the Stores had enough of PartB, but not of PartC, neither PartB or PartC would be consumed (until both were available in sufficient quantities). In addition, all Spend requirements must first be met before any Deposit interactions are carried out.

**Note:** You can create multiple Resource interactions for the same Resource Store. For example, you might want to Borrow a certain quantity, and Consume (Spend) a different quantity.

**Elements That Can Interact With Resources**

The types of elements that can interact with Resources are summarized in the next section.

Only certain kinds of elements in GoldSim can interact with Resources. The table below summarizes which elements can use Resources, and how they can use them:

---

**Elements That Can Interact With Resources**

<table>
<thead>
<tr>
<th>Element Type</th>
<th>Can Interact With Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process</td>
<td>Yes</td>
</tr>
<tr>
<td>Store</td>
<td>Yes</td>
</tr>
<tr>
<td>Resource</td>
<td>Yes</td>
</tr>
<tr>
<td>Agent</td>
<td>Yes</td>
</tr>
<tr>
<td>Function</td>
<td>Yes</td>
</tr>
</tbody>
</table>

---
### Using Resources

#### Chapter 10: Advanced Modeling Concepts

<table>
<thead>
<tr>
<th>Element</th>
<th>Resource Store Interaction</th>
<th>Allowed Usage</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triggered Event</td>
<td>Interacts when triggered.</td>
<td>Spend (discrete)</td>
<td>If triggered with a Spend Requirement and Resource is not available, trigger is held (it waits) for Resource.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deposit (discrete)</td>
<td></td>
</tr>
<tr>
<td>Event Delay</td>
<td>Interacts when triggered.</td>
<td>Spend (discrete)</td>
<td>If Requirement is Spend or Borrow, and Resource is not available, the signal is added to the queue for the element.</td>
</tr>
<tr>
<td></td>
<td>Borrowed items are returned when signal is released.</td>
<td>Borrow (discrete)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deposit (discrete)</td>
<td></td>
</tr>
<tr>
<td>Discrete Change Delay</td>
<td>Interacts when discrete change signal is received. Borrowed items are returned when signal is released.</td>
<td>Spend (discrete)</td>
<td>If Requirement is Spend or Borrow, and Resource is not available, the signal is added to the queue for the element.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Borrow (discrete)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deposit (discrete)</td>
<td></td>
</tr>
<tr>
<td>Conditional Container</td>
<td>Interacts when Activation is triggered. Borrowed items are returned when Container is deactivated.</td>
<td>Spend (discrete)</td>
<td>If triggered with a Spend or Borrow Requirement and Resource is not available, trigger is held (it waits) for Resource.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Borrow (discrete)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deposit (discrete)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Interacts while Container is Active</td>
<td>Spend Rate</td>
<td>While Container is active, spends or borrows at specified rate. If Spend Rate is specified and Resource is not available, the Container deactivates until the Resource becomes available.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deposit Rate</td>
<td></td>
</tr>
</tbody>
</table>

Specialized elements in the GoldSim Reliability Module can also interact with Resources.

**Read more:** Specifying Resources for a Triggered Event (page 369); Specifying Resources for an Event Delay (page 382); Specifying Resources for a Discrete Change Delay (page 396); Specifying Resources for a Conditional Container (page 907).

In some cases, a Resource Store may have competing requirements. For example, perhaps 10 kg of a particular Resource exists in a Store, and two Triggered Events are triggered at the same time, each one having a Discrete Spend Resource Requirement of 6 kg. How does GoldSim decide which element receives the requested Resource?

GoldSim uses the causality sequence to determine how Resources are allocated amongst competing requirements.

When you build a model, GoldSim automatically sequences the elements in the order that they must be computed. For example if A was a function of B, and B was a function of C, C would be sequenced first, followed by B, followed by A. This is referred to as the causality sequence. Although in this simple example, the sequence is obvious, for complex models with looping logic, the causality sequence...
sequence may not be readily apparent. The causality sequence can be viewed by selecting Model|Causality Sequence from the main menu.

Read more: The Causality Sequence and Element Updating (page 343).

GoldSim also provides a way to manipulate the causality sequence so that elements are computed in a specified order.

Read more: Addressing Ambiguous Causality Sequences (page 971).

In some situations, when defining a Resource Requirement Quantity/Rate, or any other property of an element that is interacting with Resources (e.g., a trigger), you may want to make the input a function of the current state of one or more Resource Stores.

GoldSim provides four locally available properties for each Resource Store available to an element:

- **ResourceName_Available** (e.g., Fuel_Available): The quantity of the specified Resource Store that is currently available.
- **ResourceName_Borrowed** (e.g., Fuel_Borrowed): The quantity of the specified Resource Store that is currently borrowed.
- **ResourceName_Spent** (e.g., Fuel_Spent): The quantity of the specified Resource Store that has been spent.
- **ResourceName_Total** (e.g., Fuel_Total): The sum of ResourceName_Available and ResourceName_Borrowed.

Read more: Understanding Locally Available Properties (page 808).

Like all locally available properties, these are accessed via the Insert Link dialog, in this case by first right-clicking inside an input field of an element that can use Resources, and then expanding the Available Properties folder (displaying all of the locally available properties that can be accessed from that location):

Like all locally available properties, these appear in input fields with the prefix “~” (e.g., ~Fuel_Available):
Because 1) an element can only interact with Resource Stores that are above it or at the same level within its Container path, and 2) if a Resource Type has multiple Stores within a particular Container path, the element can only access the Store that is closest to it, situations might arise where you may want to move Resources between Stores.

For example, if you had a Local Store of “Fuel”, as well as a Global Store of “Fuel”, if the Local Store was exhausted and the Global Store still had available Resources, elements would not be able to access the Global Store in order to meet their requirements. As long as a Local Store exists that is available to the elements, this would be the only Store that could be accessed by those elements. To access the Global Store, therefore, you would need to move Resources from the Global Store to the Local Store.

This can be done, but you must manually program how and when this takes place. The recommended way to do this is to use a combination of a Triggered Event and a Discrete Change element to remove the Resource from one Store (e.g., the Global Store) and deposit it in the other Store (e.g., the Local Store).

The specific steps involved in doing so are as follows:

1. Add a Triggered Event element at a location in the model such that it can access the Store from which you want to remove Resources (e.g., if you are moving from a Global Store to a Local Store, the Triggered Event element should be above the scope of the Local Store).

2. The Triggered Event element should have a Spend (Discrete) Resource Requirement that pulls Resources from the first Resource Store:

3. Add a Discrete Change element that is triggered by the Triggered Event. This element could be located either at the same location as the Triggered Event or at the location of the Store to which the Resources

Read more: Specifying Resources for a Triggered Event (page 369).
are being moved. The key point is that the Value for the Discrete Change should be the same as the Quantity that was specified for the Resource Requirement for the Triggered Event:

4. Within the Store to which the Resource is being moved, specify a Discrete Addition that is the output from the Discrete Change element:

Read more: Generating Resources (page 855).

A view of this structure is shown below:
Note: You cannot send (add) Resources to an inactive Container. If the Container is inactive, the Addition will be ignored.

Note: If the process of moving the Resource takes time, you could represent this by adding a Discrete Change Delay after the Discrete Change element.

**Generating Resources**

When using Resources in your model, you will define a number of events or processes that consume those Resources. Generally, you will specify an Initial Quantity when defining the Resource Store.

In some cases, however, you may want to add Resources to a Store during a simulation. GoldSim provides two ways to do this:

- Within the Store, add a Rate of Addition or Discrete Additions.

  ![Rate of Addition](image.png)

- Specify a Deposit Rate or a Discrete Deposit when defining a Resource Interaction.

The first of these two options is perhaps the most powerful, as you have great flexibility in how you can add resources. Note that the Discrete Additions field also accepts discrete change signals that have a Replace instruction. This, for example, would allow you to reset a Store to a pre-specified level (e.g., its capacity) regardless of its current level.

**Read more:** Creating and Editing Global Resource Stores (page 842); Creating and Editing Local Resource Stores (page 844).

All elements that can interact with a Resource Store can add (deposit) rather than spend Resources.

**Read more:** Elements that Can Interact With Resources (page 850).

After you have added a number of Resource Stores and elements that interact with those Stores to your model, it is often useful to be able to summarize the locations and users of the various Resources that exist in your model.

GoldSim provides several ways to do this. Within the Resources Manager (accessed via Model/Resources... or by pressing F8), a Locations button is provided:
This button lists all of the Resource Stores in the model, along with all of the elements that interact with them:

In this example, the model has 3 Resources defined. Resource1 has a Global and a Local Store; Resource2 has Local Store only; Resource3 has a Global Store only. Resource3 does not have any elements that interact with it.

Selecting an element in the tree that uses a Resource and pressing Go to element jumps to that element. The Collapse All button collapses the tree to
the top level (the Resources). (The button then becomes Expand All). Pressing Ctrl-A also toggles between expanding and collapsing the tree.

The Resources tab for a Container that has Local Stores also has a Locations button:

![Diagram](image1)

This button lists all of the Resource Stores in the Container, along with all of the elements that interact with them:

![Diagram](image2)

This is a screenshot of the Resources tab for a Container named Container1. The Resources tab includes a table with columns for ID, Units, Order, and Description. The table lists two resources: Resource1 and Resource2.

Resource1 has an initial quantity of 1000 kg, a rate of addition of 0.0 kg/day, and an event trigger with a spend quantity of 6 kg.

Resource2 has an initial quantity of 500 kg, a rate of addition of 0.0 kg/day, an event trigger, and a process named Process_X.

The Locations button is highlighted, indicating that the locations of the resources are being displayed. The diagram shows the requirements for the stores in Container1, including event triggers and process interactions.
Selecting an element in the tree that uses a Resource and pressing **Go to element** jumps to that element. The **Collapse All** button collapses the tree to the top level (the Resource Stores). (The button then becomes **Expand All**). Pressing **Ctrl-A** also toggles between expanding and collapsing the tree.

In the dialog for defining a Resource Store (either globally or locally), there is a **Users** button:

![Edit Resource Type dialog](image)

This button lists all of the users (elements) that interact with that particular Resource Store:

![Requirements for Store 'Resource1' in 'Model Root'](image)
In this example, this Resource Store (Resource1) has one element that interacts with it (Event_B).

Viewing Resource Results

After you have run a model that uses Resources, you will often want to view how those Resources are being used as a function of time.

GoldSim provides several ways to do this. Within the Resources Manager (accessed via Model | Resources… or by pressing F8), or from the Resources tab of a Container if a Local Store has been defined, a Histories button is available when the model is in Result Mode:

This button provides a history (in tabular form) of all of the Resource Stores in the model, along with all of the elements that interact with them (identifying, for example, how much of each Resource Store has been spent, borrowed or deposited):

The scroll-bar allows you to scroll horizontally to view all of the time points. The Collapse All button collapses the tree to the top level (the Resources). (The button then becomes Expand All). Pressing Ctrl-A also toggles between expanding and collapsing the tree.
**Script Elements**

**Note:** Deposits (additions) of a Resource appear as negative numbers in this table.

If you have carried out multiple realizations, additional options will be available at the bottom of the Resource Histories dialog. In particular, you must select how you want to view the Monte Carlo results:

![View Options](image)

There are four choices:

- You can view a specified realization;
- You can view the minimum value at each time point;
- You can view the maximum value at each time point; or
- You can view the mean value at each time point.

**Note:** The Histories button is grayed out when the model is in Scenario Mode. That is, you cannot view these results when running and comparing scenarios.

*Read more:* [Creating, Running and Comparing Scenarios](page 504).

You can also view time history charts of the amount of each Resource Store available by creating Expression elements that directly reference the locally-available variables for each Resource:

![Expression Properties](image)

*Read more:* [Referencing Resource Availability and Use in Input Expressions](page 852).

**Script Elements**

In some situations, you may wish to define a complex function which cannot be readily implemented using the expression editing features supplied by GoldSim.
For example, calculation of an output may require very complex logic which would be cumbersome to represent using a Selector element, or it may require a numerical solution technique (e.g., iteration); or perhaps you need to construct an array using complex logic.

GoldSim provides two separate ways to deal with such situations:

- You can develop separate program modules (written in C, C++, FORTRAN or other compatible programming languages) which can then be directly coupled with the main GoldSim algorithms. These user-defined modules are referred to here as external functions, and the elements through which they are coupled to GoldSim are called External (DLL) elements.

- You can specify a sequence of statements directly within the GoldSim interface using a Script element.

Read more: External (DLL) Elements (page 931).

Although, for some complex calculations, an External (DLL) element may be required, for many applications a Script element can be used. The key advantages of a Script over an External (DLL) is that 1) it does not require use of a separate programming language and interface; and 2) it is much more transparent (all of the “code” can be seen directly in GoldSim).

Scripts are created by inserting and editing statements or statement blocks, which may be variable definition statements, variable assignment statements, statements controlling the sequence of execution in the script (e.g., loops and if statements), or statements used for writing messages to the Run Log. The Script element sequentially evaluates the specified sequence of locally defined statements to determine its output(s).

Note: The Script element does not expect the user to learn or be familiar with a particular language. As a result, scripts are not created using a text editor. Rather, statements are inserted and edited within a “controlled environment” within the element’s property dialog in which the user selects from a number of available statement types. The syntax is already defined for each type of statement – the user simply specifies the attributes and properties for each statement via a dialog box when the statement is inserted. Statements can subsequently be moved, deleted, and edited.

To provide a quick idea of what a script looks like, a simple example in which a script is used to define a vector is shown below:

<table>
<thead>
<tr>
<th>Script</th>
<th>Statement List</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>// Defines items in a vector as a function of the index</td>
</tr>
<tr>
<td>2</td>
<td>DO index = 1, 12, 1</td>
</tr>
<tr>
<td>3</td>
<td>IF (~Result[~index]&lt;9) THEN</td>
</tr>
<tr>
<td>4</td>
<td>Result[~index] = ~index</td>
</tr>
<tr>
<td>5</td>
<td>ELSE IF (~Result[~index]&lt;9)</td>
</tr>
<tr>
<td>6</td>
<td>Result[~index] = A</td>
</tr>
<tr>
<td>7</td>
<td>ELSE</td>
</tr>
<tr>
<td>8</td>
<td>Result[~index] = B</td>
</tr>
<tr>
<td>9</td>
<td>END IF</td>
</tr>
<tr>
<td>10</td>
<td>END DO</td>
</tr>
</tbody>
</table>
The sections below discuss the use of Script elements in detail. The Script element is powerful, and, as a result, somewhat complex. It is strongly recommended that you read through these sections carefully before attempting to build your own scripts.

**Note:** In the sections that follow, the term “Script” (capitalized) refers to the element itself; the term “script” (lower case) refers to the list of statements defined by the user within the element.

### Getting Started with the Script Element

The sections below help you get started with the Script element by introducing some basic concepts, and walking through the steps required to build your first script. Before you can learn the advanced Script element features, it is essential that you first understand these basic concepts.

**The Script Element Dialog**

The properties dialog for a Script element looks like this:

Like all elements, you first specify an **Element ID** and a **Description**.

By default, all Script elements have a single primary output. When defining a Script, therefore, the first step is to define the attributes (units, type, order) of this primary output.
The **Display Units** determine the dimensions of the output. You can specify the type and order by pressing the **Type** button. The output can be defined as either a value or a condition, and can be specified as a scalar, a vector or a matrix. By default, the primary output of a new Script element is a scalar, dimensionless value.

**Read more:** [Defining and Assigning Array Variables in a Script](page 870).

**Note:** It is also possible to define additional outputs for the element (with their own dimensions) when defining local variables within a script.

**Read more:** [Defining Local Script Variables](page 867).

The primary output of a Script element is a **state variable**. A state variable provides inertia or “memory” to a system because its value is computed based on the historical value of the element’s inputs (as opposed to only being a function of the current value of the element’s inputs). All state variables have, by definition, an initial value. This allows the output to be computed when there are no historical inputs available (e.g., at the start of simulation).

**Read more:** [Understanding State Variables in GoldSim](page 341); [Understanding the Outputs of a Script Element](page 872).

The **Initial Value** input to the Script must have the same attributes (order and dimensions) as the primary output.

**Note:** The **Initial Value** must be a number or a link from a static variable (e.g., a constant Data element or a Stochastic).

Buttons at the bottom of the dialog are used to insert, delete, move and edit statements in your script. You can also print the entire script.

**Read more:** [Editing Scripts](page 887); [Printing Scripts](page 894).

If you check the **Record CPU times in the run log** box, if the element uses more than 1 CPU seconds, a message will be written to the GoldSim run log identifying the element’s name, type (i.e., Script), the number of times it was updated, and the total CPU time used.

**Read more:** [The Run Log](page 548).

Scripts are created by inserting and editing individual statements (e.g., variable definition statements, variable assignment statements) or statement blocks (e.g., loops, if statements). The Script element sequentially evaluates the specified sequence of locally defined statements to determine its output(s).

It is important to understand that scripts are **not** created using a text editor. Rather, statements are inserted and edited within a “controlled environment” within the element’s property dialog in which the user selects from a number of available statement types. The syntax is already defined for each type of statement – the user simply specifies the attributes and properties for each statement via a dialog box when the statement is inserted. Statements can subsequently be moved, deleted, and edited.

There are three ways to insert a statement into a script:
1. Select an existing statement in the script. Press the **Insert** button at the bottom of the Script element dialog. When you press this button, the following choices appear:

![Image of Insert button with statement options]

Left-click on one of the statement types. It will be inserted _below_ the selected statement.

2. Select an existing statement in the script. Press the appropriate hot-key for the statement type (indicated in the dialog above). It will be inserted _below_ the selected statement.

   *Read more:* [Hot-keys for Quickly Creating and Editing Scripts](page 887).

3. Right-click in the statement number. This will bring up a context menu, whose first item is **Insert**. This expands to display all of the statement types shown above. Left-click on one of the statement types. It will be inserted _below_ the selected statement.

   **Note:** If you hold the **Shift** key while carrying out any of these three methods for inserting a statement, the statement will be inserted _above_ the selected statement.

   **Note:** When inserting statement blocks (e.g., If-Else statement, For loops), you can select multiple statements in an existing script (by left-clicking and dragging the line numbers), and GoldSim will “wrap” the statement block around the selected statements. Note, however, that this requires that the selection is a logically complete block of code (e.g., you could not select a portion of another loop).

   *Read more:* [Controlling Program Flow in the Script Element](page 874).

### Assigning Values to Script Variables

The most fundamental of the Script element’s statements is the Variable Assignment, in which you assign a value (or condition) to a variable (typically in the form of a mathematical expression).
Selecting “Variable Assignment” from the Script element’s Insert menu (or pressing Ctrl+A when in the Script dialog) inserts the statement in the script (below the statement that was selected when this is done), and brings up the following dialog:

![Variable Assignment dialog](image)

It is first necessary to specify which variable is being assigned a value (the **Variable ID**). By default, in a new script, there is only one variable that exists: the primary output. Within a script, this primary output is always referenced as **Result**. Hence, for a new script, this is the only choice in the drop-list:

![Variable ID](image)

After selecting the **Variable ID**, GoldSim displays the attributes of the variable (in the example below, the attributes of the primary output of the Script element):

![Variable attributes](image)

You then specify how the variable is assigned (i.e., in terms of an equation, you are defining the right-hand side of the equation). The **Definition** field is the equivalent of the **Equation** field in an Expression element; you can enter any mathematical expression using all of the functions and operators you can use in other GoldSim input fields.

Three different kinds of variables can be referenced in this field:

1. You can reference outputs that are external to the Script element;
2. You can reference the primary output itself (i.e., **Result**); and/or
3. You can reference other local variables that you define in the script.

**Note:** If the Variable you are assigning is an array, the **Definition** portion of the dialog expands to provide some additional options for defining the array.

**Read more:** [Defining and Assigning Array Variables in a Script](page 870).
In the example below, two outputs that are external to the Script element (A and B) are referenced:

\[
\text{Definition: } A + B
\]

This single statement Script element (which simply adds two numbers) would then look like this:

```
1 Result = A + B
```

**Note:** Outside of the Script element, the primary output \( \text{Result} \) would be referenced by the Script element’s name. It is only referenced as \( \text{Result} \) inside the Script element.

In the example below, the primary output itself is referenced:

\[
\text{Definition: } ^\sim\text{Result} + 1
\]

The Script element would then look like this:

```
1 Result = ^\sim\text{Result} + 1
```

In this simple Script element, every time the Script element is updated, it would increment the primary output.

Note that when \( \text{Result} \) is referenced in a statement (i.e., in the right-hand side of the equation), it is referenced using the \( ^\sim \) operator. This is because \( \text{Result} \) is a locally available property. Locally available properties are only available in specific (local) parts of the model. In this case, \( \text{Result} \) can only be referenced inside the Script element, and has no meaning outside the Script element.

When inserting a link while assigning a statement in a script, you will note that \( \text{Result} \) (and any other script variables that you may have defined) are accessible only under Available Properties, and are not listed in the same manner as other outputs:

Read more: [Understanding Locally Available Properties](#) (page 808).
Finally, in the example below, Result is defined in terms of a (previously defined) local script variable (Local2) and an output external to the Script element (A):

\[ \text{Definition: } \sim \text{Local2} + A \]

The statement in the Script element would then look like this:

\[ 3 \text{ Result} = \sim \text{Local2} + A \]

(The next section describes how local variables can be defined and added to a script).

Note that like the Result variable, when a local script variable is referenced in a statement, it must be referenced using the \(\sim\) operator, since it is a locally available property.

Of course, you can assign values to the local variables in your script (not just the primary output Result). For example, in the example below, there are two previously defined local script variables (X and Local2), and the variable X is selected as the Variable ID for the statement:

It is then defined in terms of another local script variable (Local2) and an output external to the Script element (A):

\[ \text{Definition: } \sim \text{Local2} + A \]

The statement in the Script element would then look like this:

\[ 3 \text{ X} = \sim \text{Local2} + A \]

Rather than just relying on the primary output (Result), most scripts will require local variables to do such things as hold intermediate results or act as counters for loops. Local script variables can also be specified to be additional outputs to the element (so that the Script element has outputs in addition to its primary output).
Script variables are created using a Variable Definition statement. Often a script will start out with multiple Variable Definition statements that create and initialize the local variables that are subsequently used throughout the script.

You add a local variable to a script by selecting “Variable Definition” from the Script element’s Insert menu, or by simply pressing the hot-key (Ctrl+D) from the main Script dialog. Doing so inserts the statement in the script (below the statement that was selected when this is done), and brings up the following dialog:

You must specify the ID of the local variable you are creating, along with its Display Units and Type. The Display Units determine the dimensions of the variable. You can specify the type and order by pressing the Type… button. The variable can be a value or a condition, and can be specified as a scalar, a vector or a matrix. By default, a local variable is a scalar, dimensionless value.

### Note:
If the variable you are defining is an array, the Definition portion of the dialog expands to provide some additional options for defining the array.

*Read more:* Defining and Assigning Array Variables in a Script (page 870).

The Description is displayed in tool-tips when viewing the script, and is also displayed when modifying the value of the variable using Variable Assignment statements.

The Definition field is used to assign a value to the variable. This field is identical to the Definition field for a Variable Assignment statement, in that it can reference:

1. Outputs that are external to the Script element;
2. The primary output itself (i.e., Result); and/or
3. Other local variables that you define in the script.

*Read more:* Assigning Values to Script Variables (page 864).

In many cases, however, the assignment in a Variable Definition statement will simply be a constant:
This would then look like this in the script:

```
  Script
  1    Define: d = 3 m
```

Alternatively, you could define the variable and immediately define its value as an expression (as opposed to a constant):

```
  Definition: A / 2
```

This would then look like this in the script:

```
  Script
  1    Define: d = A / 2
```

Of course, you can (and almost always will) reassign a definition to a variable later in your script (using a Variable Assignment statement).

**Note:** User-defined local script variables cannot reference themselves in a Variable Definition statement (e.g., X = −X + 1 would be invalid, as there is no existing initial value of X to reference in this case). This could, however, be done in a Variable Assignment statement (since X would have been previously defined, and hence could be referenced).

When you create a local variable, you can specify that you want it to be an output for the element by checking the **Expose as Output** checkbox (which by default is cleared). This then adds an output to the Script element.

For example, if a Script element (named Script1) had two local variables (X and Y), and only X was exposed as an output, then outputs of the Script element would look like this:

```
  [Script1]
  | X |
```

The primary output (referred to as **Result** in the script) would be referenced outside of the Script element as Script1. The secondary output X would be referenced outside of the Script element as Script1.X. The other local script variable (Y) could not be referenced outside of the Script element.

**Read more:** [Understanding the Outputs of a Script Element](#) (page 872).
Defining and Assigning Array Variables in a Script

The variables in your script can be defined as arrays (vectors or matrices). When defining a local script variable that is an array in a Variable Definition statement, or assigning a value to a script variable that is an array in a Variable Assignment statement, the **Definition** portion of the dialog expands to provide some additional options for defining the array:

If you are defining a vector, there are two choices in this drop-list:

1. If you select the first one (“Entire array”), then the **Definition** field would require the field to be a vector. For example, when defining a variable (named X) that was a vector of *Days* you could enter the following to set the entire vector to zero:

   ![Example](image1)

   In the script, it would then look like this:

   ```
   script
   1 Define X[i] = Vector(Days, 0)
   ```

2. Alternatively, if you select “Item at index”, then you must also specify the item being specified, and the **Definition** field would require the field to be a scalar. For example, you could enter the following:

   ![Example](image2)

   In the script, it would then look like this:

   ```
   script
   1 Define X[i] = 7.5
   ```

   In this case, the third item of the vector would be assigned the value 7.5.

Similarly, if you are defining a matrix, there are four choices in this drop-list:

1. If you select the first one (“Entire array”), then the **Definition** field would require the field to be a matrix. For example, when defining a variable (named Y) that was a matrix of *Regions* and *Lakes* you could enter the following to set the entire matrix to 0:

   ![Example](image3)

   In the script, it would then look like this:

   ```
   script
   1 Define Y[i,j] = Math(Regions, Lakes, 0)
   ```
2. Alternatively, if you select “Item at index”, then you must also specify
the item (row and column) being specified, and the Definition field
would require the field to be a scalar. For example, you could enter the
following:

In the script, it would then look like this:

In this case, the item in the 3rd row and 5th column of the matrix would
be assigned the value 8.2.

3. If you select “Row at index”, then you must also specify the row being
specified, and the Definition field would require the field to be a
vector. For example, you could enter the following:

In the script, it would then look like this:

In this case, the item in the 4th row would be assigned as a vector of 1’s.

4. If you select “Column at index”, then you must also specify the column
being specified, and the Definition field would require the field to be a
vector. For example, you could enter the following:

In the script, it would then look like this:

In this case, the item in the 3rd column would be assigned as a vector of
1’s.
Note: When you are specifying the “Item at index” (for a vector or matrix), or “Row at index” or “Column at index” (for a matrix), you do not have to enter a number. You can enter any scalar expression. GoldSim rounds the result to the nearest integer value. This allows you to define arrays by embedding Variable Assignment statements in loops (in which case these indices are often the loop counter variable).

Read more: Controlling Program Flow in the Script Element (page 874); Script Example: Manipulating a Matrix (page 896).

The discussion above applies to assigning values to arrays in both Variable Definition statements and Variable Assignment statements. That is, both types of statements are used to assign values. It is important to understand, however, that the behavior of the two types of statements is different when assigning values to arrays.

To understand this, consider the following excerpt from a script, containing a Variable Definition statement, as well as a Variable Assignment statement (that references a previously defined variable):

```
2 #This is a Variable Definition statement
3 Define X(2) = 4
4 #This is a Variable Assignment statement
5 Y(2) = 2
```

Both X and Y are vectors. Line 3 assigns the value 4 to the second item of vector X. Line 5 assigns the value 2 to the second item of vector Y. The question is: since these statements only partially assign values to the vector (i.e., they only assign the second item), how do the statements affect the other items in the vector (that are not explicitly referenced)? The difference is as follows:

- For a Variable Definition statement, if it does not assign all the items of the array, the remaining unassigned items are automatically assigned a value of 0 (for value data types) or false (for condition data types).
- For a Variable Assignment statement, a partial assignment of a vector or matrix has no effect on the items of the vector or matrix that were not targets of assignment. That is, since all items of the array must have been previously assigned when the variable was defined (and hence already have a value), any items of the array that are not explicitly assigned are unaffected.

By default, a Script element has a single primary output. The attributes of this output are defined in the main Script dialog.

Read more: The Script Element Dialog (page 862).

It is also possible to define additional outputs for the element (with attributes and dimensions) when defining local variables within a script.

Read more: Defining Local Script Variables (page 867).

When you create a local variable, you can specify that you want it to be an output for the element by checking the Expose as Output checkbox (which by default is cleared):
For example, if a Script element (named Script1) had two local variables (X and Y), and only X was exposed as an output, then outputs of the Script element would look like this:

![Script Element Diagram]

The primary output (referred to as \textit{Result} in the script) would be referenced outside of the Script element as Script1. The secondary output X would be referenced outside of the Script element as Script1.X. The other local script variable (Y) could not be referenced outside of the Script element.

As is the case for other GoldSim elements, you can check the checkboxes at the bottom of a Script element dialog to save “Final Values” and/or “Time Histories” of all the exposed outputs (including the primary output).

There is an important difference in the behavior of the primary output and any other exposed outputs. The primary output of a Script element is a \textit{state variable}. A state variable provides inertia or “memory” to a system because its value is computed based on the historical value of the element’s inputs (as opposed to only being a function of the current value of the element’s inputs). All state variables have, by definition, an initial value. This allows the output to be computed when there are no historical inputs available (e.g., at the start of simulation).

\textit{Read more: Understanding State Variables in GoldSim} (page 341).

This has an important implication for how the variables can be used in a script. In particular, the value of the primary output is “remembered” between updates of the element. As a result, in a script you could create a Variable Assignment such as “Result = \sim \text{Result} + 1” as the very first statement. The expression would use the \textit{previous value} of the Result when it was evaluated. This is not possible for any of the other script variables, since they require a Variable Definition statement which initializes the variable every time the Script element is updated.

\textit{Note:} User-defined local script variables cannot reference themselves in a Variable Definition statement (e.g., X = \sim X + 1 would be invalid, as there is no existing initial value of X to reference in this case).

If you would like additional outputs of the Script element to be able to “remember” their previous update (and hence behave like state variables with initial conditions), you can do so by referencing the Script output in a Previous Value element outside of the Script element (and then referencing the Previous Value inside the script).
Controlling Program Flow in the Script Element

Read more: Referencing an Output’s Previous Value (page 957); Script Example: Referencing a Previous Value Element in a Script (page 897).

A scripting language is not complete without statements controlling the flow of execution. To this end, the Script element provides a number of statements for controlling programming flow, including if-else statements that support branching of statement execution, and various looping statements to support item-by-item array assignment and iterative calculations.

In particular, the program flow statements provided by the Script element consist of the following:

- If-Else statements;
- For loops;
- Do loops;
- Repeat-Until loops;
- While loops; and
- Break and Continue statements.

These are discussed in the sections below.

If-Else Statements in Scripts

If-Else statement blocks are used to test conditions that control branching of statement execution within a script. Else-if and Else instructions within the statement block provide a significant amount of control and flexibility.

An If-Else statement block can be inserted by selecting “IF-ELSE Statement” from the Script element’s Insert menu (or pressing Ctrl+I when in the Script dialog). The block is inserted below the statement that was selected when this is done. The following dialog will be displayed:

![IF-ELSE Statement dialog](image)

The condition defaults to False. You must enter a statement which evaluates to True or False (e.g., “Etime < 10 day”). After doing so, the script will look like this:

```
1 IF Etime < 10 Day THEN
2 END IF
```

Obviously, such a statement block serves no function unless it contains some statements inside the block. To insert a statement inside the block, you would select the first part of the If-Else block (in this case, line 1), and insert a statement. In the example below, a Variable Assignment statement has been inserted:
In this case, when the Script element was updated, if Etime was less than 10 days, Result would be set to \( A + 1 \) (\( A \) is an output external to the Script element).

You can also move existing statements that are above or below the If-Else block into the block by using the buttons at the bottom of the Script dialog.

Read more: Moving Statements in Scripts (page 889).

Note: If multiple statements are selected (by left-clicking and dragging the line numbers) when the If-Else statement is first inserted (and the selection is a logically complete block of code), GoldSim will “wrap” the selection in the initial If statement.

In many cases, an If-Else block will have multiple branches. The simplest such case is one in which the original If condition results in one branch if it is True, and a separate branch if it is False. This can be implemented by checking the Provide else clause box in the If-Else dialog. In the example below, an else clause (and a corresponding Variable Assignment) has been added:

(The Variable Assignment statement could be added to the else clause by selecting the ELSE statement (line 3 in this case) and inserting the new statement, or moving it from elsewhere in the script.)

If you need more complex branching, up to four else if clauses can be inserted by clicking the Click to insert “else if” clause hyperlink on the If-Else dialog. In the example below, two clauses are inserted:

After inserting corresponding Variable Assignments for each clause, the script would look like this:
Note: Within an If-Else statement block, if and else if conditions are evaluated sequentially. As soon as a True statement is found, the statements corresponding to that clause are implemented, and the remaining clauses are skipped. Hence, if two clauses have conditions that evaluate to True, only the statements associated with the first one are implemented.

One key property of If-Else statement blocks that must be understood is that each clause creates local scopes for any variables that are defined within them. For example, consider the following script:

```
Script

1. IF (ETIME < 10 day) THEN
2.    Result = A + 1
3. ELSE IF (ETIME < 20 day)
4.    Result = B
5. ELSE IF (ETIME < 30 day)
6.    Result = C
7. ELSE
8.    Result = A + 2
9. END IF

```

In this example, the local script variable Var1 is created in the first clause. This variable is local to that clause. That is, it cannot be referenced outside of that clause (without redefining it). If you wanted to reference it throughout the entire If-Else block, you would need to define it prior to the If-Else block.

Read more: Understanding Variable Scope in a Script (page 885).

For loops are one of four different types of loops provided in the Script element. They are used to loop through a set of statements. Common applications are defining and manipulating arrays, and doing an iterative calculation. For loops will be familiar to C/C++ programmers (although you certainly do not need to be familiar with these languages to use them). They require an initial integer value for the loop variable, a “Loop while true” condition, and an integer defining how the loop variable is incremented.

A For loop can be inserted by selecting “FOR Loop” from the Script element’s Insert menu (or pressing Ctrl+F when in the Script dialog). The block is inserted below the statement that was selected when this is done. The following dialog will be displayed:

---

*For Loops in Scripts*
The first field is the name of the loop variable. This becomes a local variable within the loop (identical to having used a Variable Definition statement to create a variable).

Note: This cannot be the name of an existing script variable. You must define a new variable here.

The remaining three fields are expression fields (i.e., they can be defined as an expression):

- **Start Value**. This is the initial value of the loop variable. It must be specified as a dimensionless scalar value. It is rounded to the nearest integer value.

- **Loop while true**. This must be specified as a scalar condition. This condition is evaluated at the beginning of each loop (including the first loop). If it is True, the loop is executed. If it is false, control drops out the bottom of the loop.

- **Increment by**. After a loop is completed, if the “Loop while true” condition is True, the loop variable is incremented by this value before returning to the top of the loop. It must be specified as a dimensionless scalar value (it can be positive or negative). It is rounded to the nearest integer value.

Note: If you do not change the default settings, the loop will be carried out 10 times.

Note: When the loop variable is referenced (either in the loop definition or any statements that are subsequently added to the contents of the loop), it is referenced using the ~ operator. This is because the loop variable, like all local script variables, is a locally available property. Locally available properties are only available in specific (local) parts of the model.

Read more: [Understanding Locally Available Properties](page 808).

After defining the properties of the For loop, the script will look like this:
Note that the first line indicates the start value, loop while true condition, and the increment.

Obviously, such a loop serves no function unless it contains some statements inside the looping block. To insert a statement inside the block, you would select the first part of the block (in this case, line 1), and insert a statement. In the example below, a Variable Assignment statement has been inserted:

<table>
<thead>
<tr>
<th>Line</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FOR (Var1 = 1; Var1 &lt;= 10; Var1 = Var1 + 1)</td>
</tr>
<tr>
<td>2</td>
<td>Result = Result + 1</td>
</tr>
<tr>
<td>3</td>
<td>END FOR</td>
</tr>
</tbody>
</table>

In this case, every time the Script element was updated, Result would be incremented by 1.

You can also move existing statements that are above or below the loop into the loop by using the buttons at the bottom of the Script dialog.

Read more: Moving Statements in Scripts (page 889).

**Note:** If one or more statements are selected (by left-clicking and dragging the line numbers) when the For loop is first inserted (and the selection is a logically complete block of code), GoldSim will “wrap” the selection in loop.

Because the loop variable is simply a local script variable, it can also be assigned values within the loop (i.e., in addition to being incremented at the end of each loop). For example, this loop would be interrupted if \( A = B \):

<table>
<thead>
<tr>
<th>Line</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FOR (Var1 = 1; Var1 &lt;= 10; Var1 = Var1 + 1)</td>
</tr>
<tr>
<td>2</td>
<td>Result = Result + 1</td>
</tr>
<tr>
<td>3</td>
<td>Var1 = (Var1 &lt; 11) ? Var1 + 1 : Var1</td>
</tr>
<tr>
<td>4</td>
<td>END FOR</td>
</tr>
</tbody>
</table>

For loops can have any number of statements within them. For example, a For loop could contain other nested loops, as well as complex if, then logic. Here is an example of a For loop with a nested For loop and an If-Else block:

<table>
<thead>
<tr>
<th>Line</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FOR (X = 1; X &lt;= 100; X = X + 1)</td>
</tr>
<tr>
<td>2</td>
<td>FOR (Y = 1; Y &lt;= 20; Y = Y + 1)</td>
</tr>
<tr>
<td>3</td>
<td>IF (X &gt; 50) THEN</td>
</tr>
<tr>
<td>4</td>
<td>Result = -X - Y</td>
</tr>
<tr>
<td>5</td>
<td>ELSE</td>
</tr>
<tr>
<td>6</td>
<td>Result = X + Y</td>
</tr>
<tr>
<td>7</td>
<td>END IF</td>
</tr>
<tr>
<td>8</td>
<td>END FOR</td>
</tr>
<tr>
<td>9</td>
<td>END FOR</td>
</tr>
</tbody>
</table>

Read more: If-Else Statements in Scripts (page 874).

One key property of For loops that must be understood is that the loop creates a local scope for any variables that are defined within it. Hence, the loop variable (and any other variables defined in the loop) could not be referenced outside of the loop (without first redefining them).

Read more: Understanding Variable Scope in a Script (page 885).
Do loops are one of four different types of loops provided in the Script element. They are used to loop through a set of statements. Common applications are defining and manipulating arrays, and doing an iterative calculation. Do loops will be familiar to FORTRAN programmers (although you do not need to be familiar with that language to use them). They require an initial integer value for the loop variable, an end value for the variable and an integer defining how the loop variable is incremented.

A Do loop can be inserted by selecting “DO Loop” from the Script element’s Insert menu (or pressing Ctrl+O when in the Script dialog). The block is inserted below the statement that was selected when this is done. The following dialog will be displayed:

The first field is the name of the loop variable. This becomes a local variable within the loop (identical to having used a Variable Definition statement to create a variable).

**Note:** This cannot be the name of an existing script variable. You must define a new variable here.

The remaining three fields are expression fields (i.e., they can be defined as an expression):

- **Start Value.** This is the initial value of the loop variable. It must be specified as a dimensionless scalar value. It is rounded to the nearest integer value.

- **End Value.** This is the end value of the loop variable. It must be specified as a dimensionless scalar value. It is rounded to the nearest integer value.

- **Increment by.** This is the increment of the loop variable. It must be specified as a dimensionless scalar value. It is rounded to the nearest integer value.

At the beginning of the first loop, the total number of loops to be carried out is computed as follows:

\[(\text{End Value} - \text{Start Value} + \text{Increment by})/\text{Increment by}\]

Three points should be noted here:

- If the number of loops computed above is negative, no loops are carried out;

- If **Increment by** is positive and **Start Value** > **End Value**, no loops are carried out; and
- If **Increment by** is negative and **Start Value < End Value**, no loops are carried out.

**Note:** Because the number of loops is computed at the beginning of the first loop, changing the Loop Variable in the middle of the loop has no impact on the number of loops carried out.

**Note:** If you do not change the default settings, the loop will be carried out 10 times.

**Note:** When the loop variable is referenced (i.e., in any statements that are subsequently added to the contents of the loop), it is referenced using the `~` operator. This is because the loop variable, like all local script variables, is a locally available property. Locally available properties are only available in “locally available” parts of the model.

**Read more:** [Understanding Locally Available Properties](page 808).

After defining the properties of the Do loop, the script will look like this:

```plaintext
DO Var1 = 1, 10, 1
END DO
```

Note that the first line indicates the start value, the end value, and the increment. Obviously, such a loop serves no function unless it contains some statements inside the looping block. To insert a statement inside the block, you would select the first part of the block (in this case, line 1), and insert a statement. In the example below, a Variable Assignment statement has been inserted:

```plaintext
DO Var1 = 1, 10, 1
   Result = ~Result + 1
END DO
```

In this case, every time the Script element was updated, Result would be incremented by 10. Hence, if the model had 10 timesteps, at the end of the simulation, Result would equal 110, since the loop would be evaluated 11 times (the loop is evaluated at Etime = 0 and every subsequent timestep).

You can also move existing statements that are above or below the loop into the loop by using the buttons at the bottom of the Script dialog.

**Read more:** [Moving Statements in Scripts](page 889).

**Note:** If multiple statements are selected (by left-clicking and dragging the line numbers) when the Do loop is first inserted (and the selection is a logically complete block of code), GoldSim will “wrap” the selection in the loop.
Warning: Unlike For loops, in a Do loop, the loop variable should generally not be assigned values within the loop. If it is, it will have no effect on the number of loops since the original increment logic is restored at the bottom of each loop.

Read more: For Loops in Scripts (page 876).

Do loops can have any number of statements within them. For example, a Do loop could contain other nested loops, as well as complex if, then logic. Here is an example of a Do loop with a nested Do loop and an If-Else block:

```
1 DO X = 1, 100, 1
2 DO Y = 1, 20, 1
3 IF (~X > 50) THEN
4  Result = X - Y
5 ELSE
6  Result = X - 2*Y
7 END IF
8 END DO
9 END DO
```

Read more: If-Else Statements in Scripts (page 874).

One key property of Do loops that must be understood is that the loop creates a local scope for any variables that are defined within it. Hence, the loop variable (and any other variables defined in the loop) could not be referenced outside of the loop (without first redefining them).

Read more: Understanding Variable Scope in a Script (page 885).

**Repeat-Until Loops in Scripts**

Repeat-Until loops are one of four different types of loops provided in the Script element. They are used to loop through a set of statements. Common applications are defining and manipulating arrays, and doing an iterative calculation. Repeat-Until loops require a single input: a condition that specifies whether the loop is to continue (which is evaluated at the bottom of the loop). As a result, a Repeat-Until loop is always evaluated at least once.

A Repeat-Until loop can be inserted by selecting “REPEAT-UNTIL Loop” from the Script element’s Insert menu (or pressing Ctrl+P when in the Script dialog). The block is inserted below the statement that was selected when this is done. The following dialog will be displayed:

**Loop until true** must be specified as a scalar condition. This condition is evaluated at the end of each loop. If it is False, another loop is carried out. If it is True, control drops out the bottom of the loop.

**Note:** If the condition is initially True (the default), one loop will be carried out.

After defining the properties of the Repeat-Until loop, the script will look like this:
Repeat Until loops can have any number of statements within them. For example, a Repeat-Until loop could contain other nested loops, as well as complex if, then logic. Here is an example of a Repeat-Until loop with a nested Repeat-Until loop and an If-Else block:

Read more: **If-Else Statements in Scripts** (page 874).

One key property of Repeat-Until loops that must be understood is that the loop creates a local scope for any variables that are defined within it. Hence any variables defined in the loop could not be referenced outside of the loop (without first redefining them).

Read more: **Understanding Variable Scope in a Script** (page 885).
While loops are one of four different types of loops provided in the Script element. They are used to loop through a set of statements. Common applications are defining and manipulating arrays, and doing an iterative calculation. While loops are a common construct in a number of programming languages. They require a single input: a condition that specifies whether the loop is to continue (the condition is evaluated at the top of the loop).

A While loop can be inserted by selecting “WHILE Loop” from the Script element’s Insert menu (or pressing Ctrl+W when in the Script dialog). The block is inserted below the statement that was selected when this is done. The following dialog will be displayed:

![Image of While Loop Statement dialog]

**Loop while true** must be specified as a scalar condition. This condition is evaluated at the beginning of each loop. If it is True, another loop is carried out. If it is False, control drops out the bottom of the loop.

**Note:** If the condition is initially False (the default), no loops will be carried out at all.

After defining the properties of the While loop, the script will look like this:

```plaintext
Script
1 WHILE (false)
2 END WHILE
```

Note that the first line indicates the loop’s “while” condition.

Obviously, such a loop serves no function unless it contains some statements inside the looping block. To insert a statement inside the block, you would select the first part of the block (in this case, line 1), and insert a statement. In the example below, a Variable Assignment statement has been inserted (and a condition has been defined):

```plaintext
Script
1 Define Local = 0.0
2 WHILE (Local = 10)
3 Local = Local + 1
4 END WHILE
```

In general, the condition for the While loop will involve a local variable that is also being assigned inside the loop. In this example, when control exits the loop, the variable Local would be incremented by 10.

You can also move existing statements that are above or below the loop into the loop by using the buttons at the bottom of the Script dialog.

**Read more:** [Moving Statements in Scripts](page 889).
Note: If multiple statements are selected (by left-clicking and dragging the line numbers) when the While loop is first inserted (and the selection is a logically complete block of code), GoldSim will “wrap” the selection in the loop.

While loops can have any number of statements within them. For example, a While loop could contain other nested loops, as well as complex if, then logic. Here is an example of a While loop with a nested While loop and an If-Else block:

```
Script
1 Define X = 1
2 Define Y = 1
3 WHILE (Y = 100)
4 WHILE (Y <= 20)
5 IF (X < 50) THEN
6 Result = X * Y
7 ELSE
8 Result = X * Y
9 END IF
10 Y = Y + 1
11 END WHILE
12 X = X + 1
13 END WHILE
```

Read more: If-Else Statements in Scripts (page 874).

One key property of While loops that must be understood is that the loop creates a local scope for any variables that are defined within it. Hence any variables defined in the loop can not be referenced outside of the loop (without first redefining them).

Read more: Understanding Variable Scope in a Script (page 885).

Break and Continue Statements in Scripts

Break and Continue statements are used within loops (For, Do, While and Repeat-Until) to redirect script execution. These statements can only be used inside a loop. Outside of a loop, they have no meaning and will generate an error.

A Break statement can be inserted by selecting “BREAK Statement” from the Script element’s Insert menu (or pressing Ctrl+B when in the Script dialog). A Continue statement can be inserted by selecting “CONTINUE Statement” from the Script element’s Insert menu (or pressing Ctrl+T when in the Script dialog). There are no inputs for a Break or Continue statement; hence there is no dialog. When you insert one of these statements, they are simply added to the script.

Break statements break out of the loop. That is, they redirect execution to the statement following the statement within which they are contained. Hence, Break statements are usually placed within an if statement inside of a loop to conditionally exit the loop earlier than planned.

For example, in the script below, when ETime is equal to 0 (i.e., the first time the element is updated), the loop will be broken on the second time through (i.e., when n = 2). As a result, Result will only be incremented once.
Note: A Break statement within nested loops breaks the nearest enclosing loop

Continue statements transfer control to the bottom of the enclosing loop. That is, they do not break out of the loop completely, they simply skip the statements between the Continue statement and the bottom of the loop.

For example, in the script below, a vector (X) is being defined in a Do loop. In this case, the first 5 items of the vector are not assigned (they are skipped). Items 6 through 20, however, are assigned.

Note: A Continue statement within nested loops transfers control to the bottom of the nearest enclosing loop.

Understanding Variable Scope in a Script

In order to effectively use Script elements in GoldSim, it is important to understand the concept of variable scope.

When a variable is defined in a script (either by a Variable Definition statement, or by a Do or For loop, which implicitly define loop variables), the variable persists (and hence can be referenced) from the time it is defined to the statement that closes the scope in which the variable is defined.

The scope of script variables that are 1) not inside a loop; and 2) not inside an If-Else statement consists of all lines of the script below the definition. That is, they persist until the last statement in the script.

However, script variables defined in loops or If-Else statements do not persist throughout the remainder of the script. Their scope is limited to the the loop or If-Else clause where they are defined.

To illustrate this, let’s first consider a simple Do loop:

Within this script, there are actually two variables that have been defined: 1) n is defined implicitly by the Do loop; and 2) k is defined explicitly by a Variable Definition statement.
Definition statement. For both of these variables, the scope is the Do loop itself. The scope does not extend beyond the Do loop. Hence, neither n nor k can be referenced outside the loop. In the example, k is referenced outside of the loop, and this generates an error (line 5 is displayed in red, and the model will not run).

In order to reference k outside the loop, it would have to be redefined after the loop:

In the example above, k ceased to exist after line 4, and then was redefined at line 5. Hence, the two scopes do not overlap. Note that defining k before (and within the loop) would have generated an error. That is, the following script is invalid:

In this case, line 3 is displayed in red and marked as invalid. The reason for this error is that since k was defined in line 1, its scope is the entire script. Hence, when we try to define it again in line 3, an error is generated because the two scopes overlap. Defining the same variable twice within the same scope generates a redefinition error. In the previous example, there was no error because the scopes did not overlap.

This same logic also applies for the other looping constructs in GoldSim (For loops, While loops, and Repeat-Until loops).

Read more: Controlling Program Flow in the Script Element (page 874).

If-Else statement blocks also create local scopes. In this case, the scope of any variables inside an If-Else statement block is not the entire block; rather, each clause creates local scopes for any variables that are defined within them. For example, consider the following script:

In this example, the local script variable Var1 is created in the first clause. The scope of this variable is only that clause (lines 2 and 3). Hence, it cannot be referenced outside of that clause (without redefining it). Referencing it in the second clause generates an error.

If you wanted to reference it throughout the entire If-Else block, you would need to define it prior to the If-Else block:
Scripts are created by inserting and editing individual statements (e.g., variable definition statements, variable assignment statements) or statement blocks (e.g., loops, if statements).

Statements are inserted and edited within a “controlled environment” within the element’s property dialog in which the user selects from a number of available statement types. The syntax is already defined for each type of statement – the user simply specifies the attributes and properties for each statement via a dialog box when the statement is inserted. Statements can subsequently be moved, deleted, and edited.

*Read more:* Inserting Statements into a Script (page 863).

The active line of the script is outlined in black.

To change the active line of the script, you can use the Up and Down arrow keys, or click on a statement of the script using the mouse. Holding the Ctrl key down while pressing the Up or Down arrow jumps to the top or bottom of the script, respectively. Once a statement is highlighted, it can be opened for editing by double-clicking or pressing Alt-Enter.

A valid script statement is shown in black text. Invalid statements are shown in red text.

The sections below discuss hot-keys provided to quickly insert statements and edit scripts, as well as how to delete and move script statements.

GoldSim provides a number of hot-keys for accessing common Script element functions directly from the keyboard:

<table>
<thead>
<tr>
<th>Hot-key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl+D</td>
<td>Inserts a Variable Definition statement</td>
</tr>
<tr>
<td>Ctrl+A</td>
<td>Inserts a Variable Assignment statement</td>
</tr>
<tr>
<td>Ctrl+I</td>
<td>Inserts If-Else statement block</td>
</tr>
<tr>
<td>Ctrl+F</td>
<td>Inserts FOR loop statement block</td>
</tr>
<tr>
<td>Ctrl+O</td>
<td>Inserts DO loop statement block</td>
</tr>
<tr>
<td>Ctrl+P</td>
<td>Inserts REPEAT-UNTIL loop statement block</td>
</tr>
<tr>
<td>Ctrl+W</td>
<td>Inserts WHILE loop statement block</td>
</tr>
<tr>
<td>Ctrl+M</td>
<td>Inserts a Comments statement</td>
</tr>
<tr>
<td>Hot-key</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Ctrl+L</td>
<td>Inserts a Log Message statement</td>
</tr>
<tr>
<td>Ctrl+N</td>
<td>Inserts a Log Warning statement</td>
</tr>
<tr>
<td>Ctrl+R</td>
<td>Inserts a Log Error statement</td>
</tr>
<tr>
<td>Ctrl+Delete</td>
<td>Deletes selected statement(s)</td>
</tr>
<tr>
<td>Ctrl+Shift+Delete</td>
<td>Deletes all statements</td>
</tr>
<tr>
<td>Cursor Up</td>
<td>Select previous statement</td>
</tr>
<tr>
<td>Cursor Down</td>
<td>Select next statement</td>
</tr>
<tr>
<td>Ctrl+Cursor Up</td>
<td>Selects first statement</td>
</tr>
<tr>
<td>Ctrl+Cursor Down</td>
<td>Selects last statement</td>
</tr>
<tr>
<td>Alt+Enter</td>
<td>Edits active statement (opens dialog)</td>
</tr>
<tr>
<td>Ctrl+Shift+Cursor Up</td>
<td>Moves active statement up by one row</td>
</tr>
<tr>
<td>Ctrl+Shift+Cursor Down</td>
<td>Moves active statement down by one row</td>
</tr>
</tbody>
</table>

In addition to these, several additional hot-keys are used for debugging scripts.

**Read more:** [Debugging Scripts](page 890).

**Note:** If you use one of the statement insertion hot-keys, the statement(s) will be inserted immediately **after** the active line of the script. If you use one of the insertion hot-key combinations **while holding down the shift key**, the inserted statement(s) will appear before rather than after the active line.

**Note:** When inserting statement blocks (e.g., If-Else statement, For loops), you can select one or more statements in an existing script (by left-clicking and dragging the line numbers), and when you use insertion hot-keys, GoldSim will “wrap” the statement block around the selected statements. Note, however, that this requires that the selection is a logically complete block of code (e.g., you could not select a portion of another loop).

---

**Deleting Statements from Scripts**

If a single line of a script is highlighted, it can be deleted by pressing **Ctrl+Delete** or using the **Delete** button at the bottom of the Script dialog. If you try to delete a single line that is part of a loop definition or an If-Else statement block, however, you will see a message similar to this:
As can be seen, the dialog asks if you want GoldSim to automatically expand the selection to delete the entire loop or if statement (Yes), or cancel the deletion (No).

If you would like to delete a contiguous block of statements, you can select them by highlighting their line numbers (by clicking and dragging). Once you have selected them this way, you can press Ctrl+Delete or use the Delete button on the Script element. Note, however, when you are deleting multiple statements, the highlighted block of statements for deletion must be a logically complete block of statements. For example, you could not delete just a portion of a For loop or If-Else statement block.

**Note:** Ctrl-Shift-Delete will delete all statements in a script.

### Moving Statements in Scripts

One or more statements can be moved by highlighting them and pressing Ctrl+Shift+Up or Ctrl+Shift+Down, or by using the Move Up or Move Down buttons at the bottom of the Script dialog.

There are, however, several rules that determine when GoldSim will allow a given selection to be moved. A statement that opens a scope, such as an IF statement or DO loop, cannot be moved to below the statement that closes the scope (e.g., the corresponding ELSE, ELSE IF, of End IF statement for an IF or the END DO for a DO loop). Similarly, a statement that closes a scope cannot be moved up to above the statement that opens a scope.

**Read more:** Understanding Variable Scope in a Script (page 885).

For example, in the script shown below, the line 1 could not be moved below line 3 because doing so would move a scope-opening statement below the corresponding scope-closing statement. Similarly, line 3 could not be moved above line 1.

```
1 DO Var1 = 1, 10, 1
2 Result = Result + 1
3 END DO
```

A selection of an incomplete scope, such as one that includes a DO statement without the corresponding END DO statement, cannot be moved into a higher scope or a lower nested scope, as this would create an illogical script where the scope-opening statement becomes separated from its corresponding scope-closing statement. For example, in the script shown below, line 2 cannot be moved up and line 4 cannot be moved down, but if you selected lines 2 through 4 at the same time, you would be able to move the entire Do loop statement block.

```
1 DO X = 1, 100, 1
2 DO Y = 1, 20, 1
3 Result = (X, Y) = f((X, Y), 1, 2)
4 END DO
5 END DO
```

If you select an incomplete scope and attempt to move it up/down into another scope that is at the same level, the selection will be moved up/down past the other scope, which will have the result that the scopes will be nested. As an example of this, consider the following script:
If you try to move line 4 up, it is moved all the way to the top, resulting in nested Do loops:

```
DO Var2 = 1, 10, 1
   DO Var1 = 1, 10, 1
      Result = Result + 1
      END DO
   END DO
END DO
```

**Documenting Scripts**

Particularly for complex scripts, it is very important that you document them internally. Comment statements provide an effective way to do this.

Comments can be inserted throughout scripts to document the statements and logic.

Comments can be inserted by selecting “COMMENT statement” from the Script element’s Insert menu (or pressing Ctrl+M when in the Script dialog). The comment is inserted below the statement that was selected when this is done. The following dialog will be displayed:

![Comment Statement dialog](image)

Comments can be up to 100 characters long. They appear in the script as green text, prefaced by a double forward slash:

```
DO Var2 = 1, 10, 1
   DO Var1 = 1, 10, 1
      // This is a comment
   END DO
END DO
```

It should also be pointed out that the Description entered when a local variable is created in a script also serves as an excellent documentation tool. These descriptions are shown in tool-tips when you hover over a Variable Definition or Variable Assignment statement:

![Variable Descriptions](image)

**Debugging Scripts**

In order to create and test scripts (particularly those with complex branching and looping), it is necessary to be able to step through the script to debug it.
GoldSim facilitates this by allowing you to create **breakpoints** in your script. A breakpoint provides a mechanism for pausing a simulation in the middle of a script, and subsequently allowing you to step forward (line by line), abort the calculation, or resume the calculation.

To insert a breakpoint in a script, select (highlight) the line where you would like to insert the breakpoint and press F9, or right-click and select **Breakpoint**. (You can remove a breakpoint by pressing F9, or right-clicking and selecting **Breakpoint** again). Lines with breakpoints have a red background in the script statement list. In the example below, there is a single breakpoint inserted (at line 3):

```
1  Define: RootGuess = Val
2  Define: MaxIter = 10
3  FOR (i=1; i<=MaxIter; i+=1)
4    Define: F = Val - -RootGuess*2
5    Define: FPrime = 2 * -RootGuess
6    Log Message: Iterations = (i) Root = (-RootGuess). Error = (-F)
7    IF (Abs(-F/Val) < 1e-8) THEN
8      BREAK
9    ELSE (i++) = MaxIter
10   Log Error: Sg_ Root failed to converge in f-8 loops.
11  END IF
12  // Assign the next estimate for the root
13  RootGuess = -RootGuess / -FPrime
14  END FOR
15  Result = -RootGuess
```

When you run the model, at the point when a breakpoint is reached, the model will pause, and the Script element will be opened.

When a script is paused at a breakpoint, you will notice that the buttons at the bottom of the Script dialog differ from those that are displayed when the model is in Edit Mode:

The next line to be evaluated is outlined in black. Hence, when you first reach a breakpoint, the line where the breakpoint was inserted will be outlined in black. Note that a line that is outlined in black has not yet been evaluated. The calculation is actually paused immediately prior to that line.

From the paused state in a Script element you can do the following to debug the script:

- You can hover your mouse over variables in the script to see their current values. Note that if you hover over an If statement (or a While or Repeat-Until statement), it will display the statement condition (True or False). When you hover over a For or Do loop, it will display the loop counter.

When hovering over lines to see values and debug the script, it is important to recall that a line that is outlined in black (because you have broken at that line or stepped to that line), has not yet been
evaluated. That is, an outlined line indicates that execution is paused immediately prior to the line.

- You can press the Resume button or press F5 and execution will proceed from where it is paused (until it reaches a breakpoint again).
- You can abort the model run by pressing the Abort Run button or pressing Shift+F5.
- You can turn off a breakpoint by highlighting it and pressing the Toggle Breakpoints button or pressing F9.
- You can add a new breakpoint by highlighting a line and pressing the Toggle Breakpoints button or pressing F9.
- You can press the Step button or press F10 to step through execution one line at a time.

Breakpoints are not saved with the model. That is, once you close the model file, all breakpoints are intentionally removed. You will have to re-insert them if your debugging takes place over multiple GoldSim sessions.

In some cases, in order to fully understand and debug a script, you may want to print it out. You can do this from a button on the Script dialog.

*Read more:* Printing Scripts (page 894).

Log statements, which allow results and text to be written to the model Run Log, are also useful for debugging. These are discussed in the next section.

Log statements allow you to write text and computed results to the model’s Run Log. This can also be useful for debugging scripts. The content can be a simple message, a warning, or a fatal error.

*Read more:* The Run Log (page 548).

There are three types of Log statements. All three can be inserted by selecting the appropriate type from the Script element’s Insert menu (or pressing the appropriate hot-key when in the Script dialog):

<table>
<thead>
<tr>
<th>Log Type</th>
<th>Hot-Key</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message</td>
<td>Ctrl+L</td>
<td>Writes a message to the Run Log; however, no warning is displayed after the run.</td>
</tr>
<tr>
<td>Warning</td>
<td>Ctrl+N</td>
<td>Writes a message to the Run Log; a warning is displayed after the run to alert the user.</td>
</tr>
<tr>
<td>Error</td>
<td>Ctrl+R</td>
<td>Writes a message to the Run Log; treated as a fatal error (the simulation is stopped).</td>
</tr>
</tbody>
</table>

The Log statement is inserted below the statement that was highlighted at the time the new statement was selected for insertion. The following dialog will be displayed:
Note that at the bottom of the dialog you can modify the type of Log statement (Message, Warning or Error). If you check the **Show Message** checkbox, the message will also appear in a pop-up window during the simulation whenever the statement is executed.

One of the most powerful features of the Log statement is that you can write the value of a variable, output or expression in the message. To include a scalar value in the log, type `{=VALUE}`, where `VALUE` is the output, variable or expression whose value you wish to display.

For example, assume you had a script variable named `X`, and an external output (an output from outside the Script that was referenced inside the Script) named `Y`. A Log message statement like this:

```
The value of X is =X; the value of Y is =Y.
```

would look like this in the script:

```
Define: X = 5
Define: Var1 = Y
Show & Log Message: The value of X is =X; the value of Y is =Y
```

**Note:** When referencing script variables in a Log statement, they must be referenced using the ~ (since they are locally available properties).

The message that would be displayed in a pop-up would look like this:
In the Run Log, it would look like this:

```
realization 1
0 day:
\{Script1: Script Message: The value of X is 5; the value of Y is 3
```

If the variables for which you are writing values have units, the units will be displayed. For example, if X was a length, this would be indicated in the message:

```
GoldSim

\{The value of X is 1.45 m; the value of Y is 3
```

However, you cannot control the units in which they are displayed. They are always displayed in the base units for the particular dimension (e.g., meters for lengths, seconds for time).

**Note:** No links are created by referencing external elements in a script Log statement. Since there are no links, the causality sequence is not affected in any way by the existence of the expression or reference. This means that the previous (rather than current) value of an element’s output could potentially be shown if the referenced element follows the Script element in the causality sequence. Usually, however, this issue will not come up, as element outputs referenced in Log statements will typically have been used elsewhere in the script (and hence would come before the Script element in the causality sequence).

### Printing Scripts

In some cases, in order to debug or modify your script, or explain it to someone else, you may want to print it out. You can do so by pressing the **Print** button on the Script dialog or by creating a text file of the Script contents using the key combination `Ctrl-Shft-Alt-S`.

When you do so, however, the printed text will not simply consist of what you would see if you were to look at the Script dialog. This is because for Variable Definition statements, there is information that is not explicitly shown in the script – the type, order and units of the variable. You must open the dialog for that statement to see this information.

Hence, when you print the script, GoldSim adds this information to the printed text. For example, consider this very simple script:
Script Examples

Script Example: Newton’s Method

This simple example illustrates how the Script element can be used to carry out an iterative calculation. It can be found in a file (Script.gsm) that is located in the General Examples folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu). The example uses Newton’s method to calculate the square root of 101. While you’re certainly not likely to need to calculate the square root of 101 by Newton’s method, the model does provide a nice example of how to iterate within a script.

Newton’s method is a successive approximation method for finding real roots of differentiable functions.

For a function f(X), which in this case represents the error in the approximation, which we wish to go to zero, the formula for Newton’s method is:

\[ X_{n+1} = X_n - \frac{f(X_n)}{f'(X_n)} \]

Where f’ (X_n) is the first derivative with respect to X, and X_n is the nth approximation of the root (X_0 is an initial guess for the root). Therefore, in our example of finding the square root of 101:
f(X) = X^2 - 101; and
f'(X) = 2X.

The script used for this simple example to solve for the square root of the variable Val is shown below:

```goldsim
1 Define: RootGuess = Val
2 Define: MaxIterat = 10
3 FOR I = 1; I <= MaxIterat; I = I + 1
4 Define: F = RootGuess^2 - Val
5 Define: FPrime = 2*RootGuess
6 Log Message: Iterations: (I), Root: (~RootGuess), Error: (~F)
7 IF (abs(100/100) < 1e-6) THEN
8 BREAK
9 ELSE IF (~F) THEN
10 Log Error: Sq.Root failed to converge in (~I) loops.
11 END IF
12 END FOR
13 RootGuess = (~RootGuess) - F/FPrime
14 END FOR
15 Result = (~RootGuess)
```

The script starts out with Variable Definition statements in lines 1 and 2. Note that when script variables are referenced on the right side of equations (e.g., in line 4), a ~ symbol is required. This is because script variables are locally available properties. In line 1, the script references an element outside this Script element called Val (a constant whose square root is being sought, 101). Note that referencing another element’s output does not require the ~ symbol.

Read more: Defining Local Script Variables (page 867); Assigning Values to Script Variables (page 864).

After the Variable Definition statements, a FOR loop is defined. The FOR loop starting in line 3 shows that it creates a loop variable (named “I”) that is initially 1 and is incremented by 1 on each iteration, continuing while the expression “I<=MaxIterat” is true. Hence, the FOR loop executes the statements it contains until it is broken by a BREAK statement, or the loop count reaches the specified maximum number of iterations possible. In each FOR loop iteration, two more Variable Definition statements (with expressions as the assignments) compute the values for the function (F) and the derivative (FPrime). A Variable Assignment statement (line 13) then generates a new approximation of the square root at the bottom of the loop.

The IF statement block beginning at line 7 tests the approximation error in the current estimate of the square root. If it is sufficiently small, a BREAK statement is executed to terminate the FOR loop.

Read more: If-Else Statements in Scripts (page 874); Break and Continue Statements in Scripts (page 884).

If the approximation error remains high and the maximum number of iterations is reached, a LOG statement (line 10) is written to the Run Log noting that the script failed to converge. Line 6 is another LOG statement reporting the progress of the successive approximations for each iteration of the FOR loop.

Read more: Logging Messages in Scripts (page 892).

Script Example: Manipulating a Matrix

A very common application of the Script element is to construct and/or manipulate arrays. This is typically facilitated by using one of the Script element’s looping constructs to assign values to rows, columns and/or individual items. The simple example described here illustrates how the Script element can be used to carry out array operations such as these. It can be found in a file...
(Script.gsm) that is located in the General Examples folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

Read more: Defining and Assigning Array Variables in a Script (page 870).

The example starts with an input matrix. In the input matrix, each column represents a variable, and each row represents an observation of the variable. The script computes the correlations among the various columns (variables) of the matrix, producing a symmetric correlation matrix (rows and columns both indexed by variable). The correlation coefficient, $C_{xy}$, between two columns of data (i.e., the correlation between two variables) is given by the following equation:

$$C_{xy} = \frac{\sum_{i=1}^{n}(x_i - m_x)(y_i - m_y)}{ns_x s_y}$$

In this equation, $x$ and $y$ are the variables, $x_i$ and $y_i$ are the $i^{th}$ observations of variable $x$ and $y$, respectively. $m_x$ and $m_y$ are the means of the observations for those variables (i.e., the means of all values in a particular column). $s_x$ and $s_y$ are the standard deviations for those observations. $n$ is the number of observations (i.e., the number of rows).

The script used for this simple example is shown below:

```
At the top of the script (lines 2 and 4), Variable Definition statements (with expressions as the assignments) compute the means and standard deviations of each column.

Read more: Defining Local Script Variables (page 867); Assigning Values to Script Variables (page 864).

The calculation of the correlation coefficient for each column pair is then carried out using two Variable Assignment statements. The first (line 11) is in the middle of three nested DO loops. The second (line 14) is at the bottom of the second nested loop.

Read more: Do Loops in Scripts (page 879).

By default, a Script element has a single primary output. However, it is possible to define additional outputs for the element (with their own attributes and dimensions) when defining local variables within a script.

Read more: Understanding the Outputs of a Script Element (page 872).
There is an important difference in the behavior of the primary output and any other exposed outputs. The primary output of a Script element is a state variable. A state variable provides inertia or “memory” to a system because its value is computed based on the historical value of the element’s inputs (as opposed to only being a function of the current value of the element’s inputs). All state variables have, by definition, an initial value. This allows the output to be computed when there are no historical inputs available (e.g., at the start of a simulation).

Read more: Understanding State Variables in GoldSim (page 341).

This has an important implication for how the variables can be used in a script. In particular, the value of the primary output is “remembered” between updates of the element. As a result, in a script you could create a Variable Assignment such as “Result = ~Result + 1” as the very first statement. The expression would use the previous value of the Result when it was evaluated. This is not possible for any of the other script variables, since they require a Variable Definition statement which initializes the variable every time the Script element is updated.

If you would like additional outputs of the Script element to be able to “remember” their previous update (and hence behave like state variables with initial conditions), you can do so by referencing the Script element’s output in a Previous Value element outside of the Script element (and then referencing the Previous Value inside the script).

Read more: Referencing an Output’s Previous Value (page 957).

This very simple example illustrates how the Script element can be used in conjunction with a Previous Value element. It can be found in a file (Script.gsm) that is located in the General Examples folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

The script simply updates the primary output (Result) and a secondary variable (Exposed) every time the Script element is updated (in this case, every timestep). In order for the defined variable (Exposed) to be “remembered” between timesteps, it is stored in a Previous Value element, and it is that element that is referenced by the script):

- Script
- 
- 
- 1
- 2
- Definition: Exposed = PV.Exposed + 1

As can be seen, the script references the Previous Value element in a loop:

Browser View of a Script Element

The browser view of a Script element is shown below:

In this example, two script variables have been added as exposed outputs (N and X). By default, the Script element would only have a single output (the primary
output, having the same name as the element). The Script element only has a single input – the Initial Value.

**Note:** Element inputs and outputs are only shown in the browser if you choose to **Show element subitems** (accessed via the browser context menu by right-clicking in the browser).

---

**Using Conditional Containers**

One of the advanced features in GoldSim is to make Containers conditional. Conditionality allows you to make a Container and all of its contents inactive until specific events occur and/or specific conditions are met.

Elements in an inactive container are “dormant”. That is, they are not updated or recalculated each timestep, and while they are inactive their output values never change. When other specific events occur and/or conditions are met, the Container (and its contents) can become active (and hence carry out their normal calculations). A conditional Container can activate and deactivate multiple times during a simulation.

Conditionality is a very powerful feature, and can be used to

- temporarily “turn off” certain parts of your model (e.g., during a testing phase); or
- simulate processes or features which themselves only exist or are active during certain parts of your simulation.

As an example of the latter, suppose that you wished to model a facility (e.g., a chemical plant) which had a contingency plan for repairs following some event (e.g., an accident). This contingency plan could be represented within a conditional Container, and the Container could then be activated when the event occurred. The Container representing the contingency plan would operate (accumulating costs and other consequences) until repairs were completed, and then deactivate (until another event occurred).

A simple example which illustrates the use of conditional Containers (ConditionalContainer.gsm) can be found in the Containers subfolder of the General Examples folder of your GoldSim directory (accessed by selecting **File | Open Example...** from the main menu). The folder also includes a more complex example of the use of conditional Containers to simulate projects (ProjectSimulation.gsm).

Before discussing how conditional Containers can be activated and deactivated, it is useful to first explain how elements behave when they are within a conditional Container.

Elements within conditional Containers behave as follows:

- Before a Container is activated for the first time in a realization, any elements it contains behave as follows:
  - Data elements and Expressions that are static (i.e., have constant inputs or are functions of elements with constant inputs) are computed at the beginning of each realization.
Using Conditional Containers

- Static Stochastic elements (Stochastics whose inputs are constant, and have no triggers) are sampled at the beginning of each realization.

- Conditional Containers have no effect on most Time Series elements, which always output their results. Time Series elements that are set to record, however, cannot be placed inside a Conditional Container that can be deactivated.

- Outputs with clearly defined non-zero initial values (i.e., Integrators, Reservoirs, Material and Information Delays, Previous Value elements, Status elements, and Milestones) take on their initial values at the beginning of each realization.

- All other outputs are set to zero.

- Once a Container is activated, all the elements that it contains output their normal (computed) output.

- After a Container is deactivated, any elements it contains behave as follows:
  - Most outputs that are continuous (as opposed to discrete) continue to transmit their last active value.
  - The exception to this are outputs that represent material flows (e.g., the output of a Material Delay; the Overflow Rate from a Reservoir). These outflows are set to zero.

Note: When a Container is triggered to deactivate, all elements inside the Container are updated before the Container deactivates.

- While a Container is inactive,
  - All incoming material flows (e.g., Rates of Addition to Reservoirs, Flows into Material Delays) or discrete changes of material are ignored and discarded. Warning messages are written to the Run Log whenever material flows from active elements (outside the conditional Container) to inactive elements are discarded.
  - Any other discrete signals sent to inactive elements are ignored and discarded. Warning messages are written to the Run Log when discrete additions or withdrawals of material flows to inactive elements are discarded.
  - Any timed events or events triggered using an On Event trigger are ignored and discarded.
  - Any discrete or continuous signals or flows that are in transit within Delays are “frozen” (the conveyor is stopped).
  - If an inactive Container is activated at exactly the same time as some child elements are triggered, the triggers to the children are not ignored (they are applied immediately upon activation of the Container).
Using Conditional Containers

**Warning:** On Changed, On True and On False events are triggered whenever the element determines that the argument to the event has changed, become true, or become false, respectively. If the element is in a conditional Container, it does not evaluate these arguments while it is inactive, and therefore some On Changed, On True and On False events that occur while the Container is inactive can be “delayed” until the element activates (rather than ignored). For example, if an On True trigger is defined as “Etime > 15 days”, and the element is inactive until 20 days, the event will actually be triggered at 20 days (since, from the element’s viewpoint, this is first time that it is able to determine that the condition has become true).

As a simple example of the behavior of an element inside a conditional Container, consider an Integrator with an Initial Value of 10, and a Rate of Change of 5 per day. Assume that the Integrator was within a conditional Container that was initially inactive, was activated at 10 days, and deactivated at 20 days.

In this case, the output of the Integrator would behave as shown below:

The Integrator starts at 10. It remains at 10 until the Integrator activates at 10 days. Between 10 and 20 days, the output increases linearly (to 60), and after 20 days, the output stays constant at 60.

Double-clicking on any Container brings up the Container’s property dialog:

---

**Enabling and Disabling Conditionality**
Containers can be made conditional by selecting the **Conditionality** feature in the Container dialog. When you do so, a **Conditionality** tab is added to the Container dialog:

You can turn off conditionality by selecting the **Conditionality** feature in the Container dialog to clear the box. When you do so, the **Conditionality** tab is removed.

Conditional Containers are represented in the graphics pane as follows (it is red instead of yellow):
Warning: When you make a Container conditional, the Treat as SubSystem feature is also automatically selected (and cannot be deselected unless you first turn off Conditionality). That is, a Conditional Container, by definition, is treated as a SubSystem. Because a Conditional Container is treated as a SubSystem, this puts certain limitations on how Conditional Containers can be used.

Read more: Treating a Container as a SubSystem (page 135).

When you enable conditionality for a Container, seven outputs are added to the Container. While a regular Container (i.e., one which is not conditional) has no outputs of its own, the seven outputs of a conditional Container are properties of the Container itself.

Four of the outputs are values or conditions:

- **Activity_Status**: This is a condition which is True when the Container is active.
- **Completion_Status**: This is a condition which is True when the Container has deactivated and the Container is considered to have been completed. A conditional Container is considered to have completed if, at the time it deactivates, the expression in the field named **On deactivation, output Completion event if this condition is true** evaluates to True. If a Container reactivates after completing, the Completion_Status is reset to False.

  Read more: Deactivating a Container (page 905).

- **Num_Activations**: This is the cumulative number of times that the Container has been activated.
- **Duration**: This represents the duration that the Container has been Active.

If you wish to reference any of these outputs outside of the Container, you must do so by referencing the Container name as a prefix (e.g., Container1.Duration).

You cannot reference the **Activity_Status** or the **Completion_Status** of a Container inside a Container (GoldSim will say that this will create a recursive system). There is, in fact, no reason to do so, since if an element in the Container is being updated, the Container must be active (so that the Activity_Status must be True and the Completion_Status must be False).

In some cases, you may want to reference the **Duration** and/or **Num_Activations** outputs inside the Container or when defining Deactivation triggers for the Container. GoldSim does support this. However, you cannot do so by using the Container name (i.e., Container1.Duration). Instead, you must reference these two variables using the prefix “~” (e.g., ~Duration and ~Num_Activations). As indicated by the way they are referenced, these two outputs are examples of locally available properties.

Read more: Understanding Locally Available Properties (page 808).

If you access the Insert Link dialog inside a conditional Container (by right-clicking inside an input field of an element within the Container or within the Deactivate trigger dialog for the Container), these two outputs appear in an...
Available Properties folder (displaying all of the locally available properties that can be accessed from that location):

Conditional Containers also have three additional outputs which are events:

**Activation_Event:** This is an event that is output when the Container is activated. If a Container is already active when an Activation trigger is received, it will not output another Activation_Event at that time.

**Read more:** Activating a Container (page 904).

**Completion_Event:** This is an event which is output when the Container has deactivated and the Container is considered to have been completed. A conditional Container is considered to have completed if, at the time it deactivates, the expression in the field named On deactivation, output Completion event if this condition is true evaluates to True.

**Read more:** Deactivating a Container (page 905).

**Termination_Event:** This is an event which is output when the Container has deactivated and the Container is considered not to have been completed. A conditional Container is considered to have not completed if, at the time it deactivates, the expression in the field named On deactivation, output Completion event if this condition is true evaluates to False.

You control when a conditional Container activates via the Activation… button on the Conditionality tab, which provides access to a standard Trigger dialog.

**Read more:** Understanding Event Triggering (page 355).

By default, the Activation is set to “Never”. This means that the Container will never activate.

**Note:** You can tell if an activation trigger has been defined from the appearance of the Trigger… button. If a trigger is defined, the rectangle next to the lightning bolt is bright green; otherwise it is dark green. And like all Trigger… buttons, it displays a tool-tip. If there is no trigger defined, the tool-tip will display “Never activates”.

You can explicitly control when a conditional Container activates by defining one or more triggers (e.g., “Activate when Time > 10 days”).

Whenever a conditional Container is activated, an Activation_Event output is released.

**Read more:** Outputs of a Conditional Container (page 903).
Note: You can record all activation and deactivation times in the model’s Run Log. This logging can be activated via the General tab of the Options dialog (accessed via Model|Options from the main menu).

Read more: The General Tab of the Options Dialog (page 446).

If you wish the Container to activate when its parent Container activates, set the Activation trigger to Auto Activate. Note that if a conditional Container is set to Auto Activate and it is not within a conditional Container, it activates when the Model (root) Container activates (i.e., at the start of the realization).

Several important points should be noted regarding activation of Containers:

- Even if the Container is activated via a trigger, it can only activate if its parent Container is active. If its parent Container is inactive, activation triggers will be ignored (and discarded).
- A Container can be activated multiple times during a realization. Of course, in order for a Container to be reactivated, it must first be deactivated. Activation triggers that occur while the Container is active are ignored (they are not “stored” or saved).
- If an inactive Container is activated at exactly the same time as some child elements are triggered, the triggers to the children are not ignored (they are applied immediately upon activation of the Container).

One common use of conditional Containers is to use the activation of a Container to simultaneously trigger multiple elements within the Container.

For example, if you wished to trigger a number of Discrete Change elements and/or resample a number of Stochastics, you could do so by placing them inside a conditional Container and specifying (in their Trigger dialogs) that they are automatically triggered (Auto Resample for Stochastics and Auto Trigger for Discrete Changes). When the Container was triggered to activate, the Stochastics would automatically be resampled, and the Discrete Changes would automatically be triggered.

Note: You cannot reference the locally available properties Duration or Num_Activations in an Activation trigger. Arguments to an Activation trigger must come from outside of the Container.

Deactivating a Container

Once a Container is activated, it can be deactivated via the Deactivation... button on the Conditionality tab, which provides access to a standard Trigger dialog.

Read more: Understanding Event Triggering (page 355).

You can explicitly control when a conditional Container deactivates by defining one or more triggers.

By default, the Deactivation is set to Auto Deactivate. This means that the Container will deactivate when its parent Container deactivates. Note that if a conditional Container is not within a conditional Container, it deactivates when the Model (root) Container deactivates (i.e., at the end of the realization).
Note: You can tell if a deactivation trigger has been defined from the appearance of the Trigger… button. If a trigger is defined, the rectangle next to the lightning bolt is bright green; otherwise it is dark green. And like all Trigger… buttons, it displays a tool-tip. If there is no trigger defined, the tool-tip will display “Automatically deactivates when parent deactivates”, where parent is the name of the parent Container.

One special trigger type is provided within the Deactivation trigger dialog:

![Deactivation Trigger Dialog]

The At Duration trigger deactivates the Container when the Duration output exceeds the specified Trigger Definition (which must represent a length of time). The Duration output represents the amount of time that the Container has been active. Hence, this provides a convenient mechanism to deactivate a Container once it has been active for a specified amount of time. The Trigger Definition input in this case would typically be defined by an element inside the Container.

Read more: Outputs of a Conditional Container (page 903).

Note: You can record all activation and deactivation times in the model’s Run Log. This logging can be activated via the General tab of the Options dialog (accessed via Model|Options from the main menu).

Read more: The General Tab of the Options Dialog (page 446).

Several points should be noted regarding deactivation of Containers:

- A Container can be deactivated multiple times during a realization. Of course, in order for a Container to be deactivated a second time, it must first be reactivated. Deactivation triggers which occur while the Container is inactive are ignored (they are not “stored” or saved).
- All conditional Containers have an Auto Deactivate trigger (that cannot be deleted). Hence, it will always immediately deactivate if its parent Container deactivates.
- A conditional Container can be activated and immediately deactivated by assigning the same trigger for Activation and Deactivation.
- When a Container is triggered to deactivate, all elements inside the Container are updated before the Container deactivates.
When a conditional Container is deactivated, one of two possible events are released, depending on the condition specified in the **Treat as completion if this condition is true** field (which defaults to True):

- **Completion_Event**: This event is output when the Container has deactivated and the Container is considered to have been *completed*. A conditional Container is considered to have completed if, at the time it deactivates, the expression in the field named **On deactivation, output Completion event if this condition is true** evaluates to True.

- **Termination_Event**: This event is output when the Container has deactivated and the Container is considered *not* to have been *completed*. A conditional Container is considered to have *not completed* if, at the time it deactivates, the expression in the field named **On deactivation, output Completion event if this condition is true** evaluates to False.

**Note**: If a Container is deactivated due to deactivation of a parent Container or due to an operating Resource Requirement not being met, neither a Completion_Event nor a Termination_Event is fired.

**Read more**: [Specifying Resources for a Conditional Container](#) (page 907).

Deactivating a Container and differentiating whether it has terminated or completed is particularly useful for simulating projects (i.e., when you are using the conditional Container to simulate a project or a task). In this case, you often need to track whether or not various tasks (or collections of tasks) have been successfully completed, as this may impact how subsequent tasks are carried out.

Most elements with triggers can be assigned an Auto Trigger. An Auto Trigger requires no user-defined Trigger Definition and its behavior is defined by its context (i.e., the type of element). Auto Triggers react to the activation or deactivation of their parent Container.

**Read more**: [Understanding Event Triggering](#) (page 355).

In most cases, an Auto Trigger is used in association with the activation of a conditional Container. In particular, if an element inside a conditional Container is given an Auto Trigger, it will be triggered when its parent Container activates.

In one instance, an Auto Trigger is associated with the deactivation of a Container. In particular, conditional Containers all have **Auto Deactivate** triggers (that cannot be deleted). This means that the Container will deactivate when its parent Container deactivates. Note that if a conditional Container is not within a conditional Container, it deactivates when the Model (root) Container deactivates (i.e., at the end of the realization).

Conditional Containers can have specified Resource Requirements.

**Read more**: [Using Resources](#) (page 839).
There are two types of Resource Requirements that can be specified for Conditional Containers:

- Requirements to activate the Container;
- Requirements for the Container to operate.

To define a Resource Requirement for activating a Conditional Container, press the Resources… button in the Activation… trigger dialog for the element. The following dialog will be displayed:

A Conditional Container Activation trigger interacts with the specified Resource Stores when the Container is activated, and can only have three types of interactions (specified in the Usage column):

- **Spend (discrete):** A discrete quantity of the Resource is required in order to activate the Container. If triggered with a Spend Requirement and the requested Resource quantity is not available, the trigger is held (it waits) until the Resource becomes available. If the Resource becomes available at a later time in the simulation, this creates a delay in the activation of the Container; if the Resource never becomes available, the Container is never activated.

- **Borrow (discrete):** A discrete quantity of the Resource is required in order to activate the Container. If triggered with a Borrow Requirement and the requested Resource quantity is not available, the trigger is held (it waits) until the Resource becomes available. If the Resource becomes available at a later time in the simulation, this creates a delay in the activation of the Container; if the Resource never becomes available, the Container is never activated. Once the Container is deactivated, the borrowed quantity is returned to the Resource Store.

- **Deposit (discrete):** A discrete quantity of the Resource is created and deposited with the Store when the Container is activated.

To define a Resource Requirement for operating a Conditional Container, press the Resources… button on the Conditionality tab of the Conditional Container (labeled “Define operating Resource Requirements”). The following dialog will be displayed:
You can add a Resource Requirement by pressing the **Add** button.

*Read more:* [Interacting with Resources](#) (page 848).

A Conditional Container Operation Requirement interacts with the specified Resource Stores when the Container is activated, and can only have two types of interactions (specified in the **Usage** column):

- **Spend At Rate:** A specified continuous spend rate of the Resource is required in order for the Container to remain Active. If the requested Resource is not available in sufficient quantity to maintain the requested spend rate, the Conditional Container is deactivated. If it becomes available again, the Container reactivates.

  *Note:* If a Container is deactivated due to an operating Resource Requirement not being met, although the Activity_Status is set to False, neither a Completion_Event nor a Termination_Event is fired.

*Read more:* [Outputs of a Conditional Container](#) (page 903).

- **Deposit At Rate:** While the Container is active, the specified deposit rate is added to the Store.

Resources are an advanced feature, and you should read the section listed below before attempting to use them.

*Read more:* [Using Resources](#) (page 839).

The browser view of a conditional Container is shown below:
Using External Application Elements

In addition to presenting the Container’s contents, a “Conditionality” folder is added to the Container’s browser view. This folder contains the input triggers for activation and other inputs, and the outputs of the Container.

**Note:** Element inputs and outputs are only shown in the browser if you choose to **Show element subitems** (accessed via the browser context menu by right-clicking in the browser).

**Read more:** Using the Browser (page 105).

**Using External Application Elements**

GoldSim provides three elements that can interact with external applications:

- Spreadsheet elements;
- External (DLL) elements; and
- File elements.

These are discussed below.

**Spreadsheet Elements**

A Spreadsheet element allows you to dynamically exchange data with a Microsoft Excel spreadsheet file. You can transmit data into specified cells in the spreadsheet, have the spreadsheet recalculate, and copy data from "result" cells back into your GoldSim model.

Alternatively, you can simply use a spreadsheet as a source of input data to GoldSim, or as a repository for output data.

**Note:** If you wish to import lookup tables or time history data from a spreadsheet into GoldSim, or export time history data from GoldSim to a spreadsheet, GoldSim provides specialized ways to do this using the Lookup Table, Time Series and Time History Result elements that do not require a Spreadsheet element.

**Read more:** Linking a Lookup Table to a Spreadsheet (page 306); Importing Data into a Time Series from a Spreadsheet (page 194); Exporting from a Time History Result Element to a Spreadsheet (page 720).
Note: GoldSim supports .xlsx, .xlsm, and .xls Excel files. However, if you have an older version of Excel (prior to Office 2007), you will need to install Microsoft's Office Compatibility Pack in order to read .xlsx and .xlsm files. Excel 2007 and later support a larger worksheet size (1,048,576 rows by 16,384 columns) than earlier versions (65,536 rows by 256 columns). If you wish to exchange data between an extended worksheet range and GoldSim, you must use Excel 2007 or newer, and the file format must be .xlsx or .xlsm. Note that GoldSim does not officially support versions of Excel prior to Excel 2003.

Warning: You should never try to open a spreadsheet that is linked to GoldSim while a simulation is running. In addition, prior to running a GoldSim model that is linked to a spreadsheet, all referenced spreadsheets should be closed. If you try to interact with a spreadsheet that is being used by GoldSim during a simulation, all interaction with the spreadsheet is blocked by GoldSim. However, under some circumstances this may not be possible and could lead to errors. Hence, as a general rule, you should not use Excel while a GoldSim model that references Excel is running.

An example model which uses a Spreadsheet element (Spreadsheet.gsm) can be found in the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

The properties dialog for a Spreadsheet element looks like this:

You must first enter the name of a Microsoft Excel spreadsheet file to which you wish to link by pressing the Options >> button. This will provide options for either selecting an existing file, or creating (and then selecting) a new file.
**Note:** If you select a file in the same directory as (or a subdirectory below) your GoldSim .gsm file, GoldSim will subsequently display just a local path. If you select a file in a directory above your .gsm file, it will display the full path.

Once you have selected a file, you can subsequently use the Options » button to select a different file. You can also use the Options» button to open the selected file in Excel. (Note that multiple Spreadsheet elements can link to the same spreadsheet file.)

Spreadsheet elements can be used in one of three ways:

1. You can use the spreadsheet as a database and import data from the spreadsheet into GoldSim at the beginning of the simulation.
2. You can use the spreadsheet as a repository for results and export data from GoldSim to the spreadsheet throughout a simulation.
3. You can use the spreadsheet as a function by exporting data into specified cells in the spreadsheet from GoldSim, having the spreadsheet recalculate, and then import data from "result" cells in the spreadsheet back into your GoldSim model.

In all cases, you need to define either at least one input to the spreadsheet element (data that you are exporting from GoldSim to the spreadsheet), or one output from the spreadsheet (data that you are importing from the spreadsheet into GoldSim).

The dialog lists all of the inputs and outputs that you have specified (inputs are listed in the table first).

You add inputs and outputs by pressing the Add... button (which only becomes available after you have specified a spreadsheet file). After pressing the Add... button, the following dialog is displayed:

![Import or Export Selection](image)

You must select whether you want to Import data from the spreadsheet (creating an output to the element) or Export data to the spreadsheet (creating an input for the element). If Use Wizard to define spreadsheet inputs and outputs is checked, a wizard will pop up to guide you through the process of creating an input or output. Otherwise, a dialog will appear. It is recommended that you use the wizard until you are familiar with the use of the Spreadsheet element.

**Read more:** Using the Spreadsheet Wizard to Define Spreadsheet Inputs (page 914); Using the Spreadsheet Wizard to Define Spreadsheet Outputs (page 921).

Once you have defined or edited an input or output, the main Spreadsheet element dialog is displayed again, listing all of the inputs and outputs that you have specified (inputs are listed in the table first):
An input or output can be edited by either selecting the row and pressing **Edit...**, or by double-clicking on the row. You can edit the spreadsheet location of any entry by selecting the row and pressing **Location...**. An input or output can be deleted by pressing the **Remove** button.

The up and down arrow buttons move the entries up and down the list (although inputs will always be listed before outputs). Note that the relative positions of the rows have no effect on how the element functions, and such repositioning is purely cosmetic.

The **Shift...** button provides access to a dialog that allows you to quickly edit the spreadsheet location of one or more inputs or outputs by shifting them by one or more sheets, rows or columns.

*Read more:* Shifting Ranges for Inputs and Outputs to a Spreadsheet Element (page 925).

If you check the **Record CPU times in the run log** box, if the element uses more than 1 CPU seconds, a message will be written to the GoldSim run log identifying the element’s name, type (i.e., Spreadsheet), the number of times it was updated, and the total CPU time used.

*Read more:* The Run Log (page 548).

If you press the **Export...** button from the Import or Export Selection dialog (and the checkbox for using the wizard is cleared), or if you select an existing input in the main Spreadsheet dialog and press the **Edit...** button, the following dialog is displayed:
This dialog is used to create an input to the Spreadsheet element. An input to a Spreadsheet element is defined as an existing GoldSim output that you wish to export from GoldSim to the spreadsheet file.

You define an input as follows:

1. Select the GoldSim output that you wish to export to the spreadsheet by pressing the **Select Output**... button, which will display a browser for selecting an output. You can select a scalar, a vector or a matrix. The output can be a Value or a Condition. True Conditions will export into the spreadsheet as 1; False Conditions will export into the spreadsheet as 0.

2. The **Units in Spreadsheet** will default to the display units for the output that you selected. If the units of the data in the spreadsheet are different from this, you should change the units accordingly.

3. The **Name** of the input will default to Input\(n\), where \(n\) is an integer. This name is used to label the inputs to the Spreadsheet element in the browser and the element’s input interface. If you wish to, you can change this name, but it is not required. The optional **Description** appears only in this dialog and is useful for helping to document your spreadsheet link.

4. You must then select the location in the spreadsheet to which you wish to export the GoldSim data. You can do this by specifying the Sheet Name and Cell Range directly. Alternatively, you can press the **Location**... button, which opens the spreadsheet and provides a dialog allowing you to directly select the desired cell range using your mouse.

---

**Note:** The size of the range must be consistent with the dimensions of the input which you are sending to the spreadsheet. Hence, a scalar input must map to a single cell; a vector input must map to a single column or row range with the number of rows or columns matching the number of items in the vector; and a matrix input must map to a rectangular range with the number of rows and columns matching the dimensions of the matrix (e.g., a matrix with 3 rows and 6 columns would have to map to a range in the spreadsheet 3 rows long and 6 columns wide).

---

If required, you can control the sheet and cell range into which GoldSim exports data dynamically. For example, you could instruct GoldSim to change the location to which it exports data based on the simulation time or the realization.

**Read more:** [Defining Offsets for Inputs and Outputs to a Spreadsheet Element](page 923).

If you would like to edit the location of several inputs (or outputs) simultaneously by shifting them by sheet, row or column, you can do this from the main Spreadsheet element dialog using the **Shift**... button.

**Read more:** [Shifting Ranges for Inputs and Outputs to a Spreadsheet Element](page 925).

If you press the **Export**... button from the Import or Export Selection dialog, and the checkbox for using the wizard is checked, the Spreadsheet Wizard will be displayed:

**Using the Spreadsheet Wizard to Define Spreadsheet Inputs**

This wizard is used to create an input to the Spreadsheet element. An input to a Spreadsheet element is defined as an existing GoldSim output that you wish to export from GoldSim to the spreadsheet file. Until you become comfortable creating inputs and outputs for Spreadsheet elements, it is recommended that you use the wizard.

The wizard has four pages:

1. On the first page you must select the GoldSim output that you wish to export to the spreadsheet. The wizard displays a browser for selecting an output. You can select a scalar, a vector or a matrix. The output can be a Value or a Condition. True Conditions will export into the spreadsheet as 1; False Conditions will export into the spreadsheet as 0.

2. On the second page, you specify the **Name** of the input. It will default to **Input n**, where n is an integer. This name is used to label the inputs to the Spreadsheet element in the browser and the element’s input interface. If you wish to, you can change this name, but it is not required. The optional **Description** appears only in the dialog for subsequently editing this Input and is useful for helping to document your spreadsheet link.

3. On the third page, you must specify the **Units in Spreadsheet**. They will default to the display units for the output that you selected. If the units of the data in the spreadsheet are different from this, you should change the units accordingly. Note that this page is skipped if the output is dimensionless or a Condition.

4. On the last page of the wizard, you select the location in the spreadsheet to which you wish to export the GoldSim data. You can do this specifying the Sheet Name and Cell Range directly. Alternatively, you can press the **Location…** button, which opens the spreadsheet and provides a dialog allowing you to directly select the desired cell range using your mouse.
Note: The size of the range must be consistent with the dimensions of the input which you are sending to the spreadsheet. Hence, a scalar input must map to a single cell; a vector input must map to a single column or row range with the number of rows or columns matching the number of items in the vector; and a matrix input must map to a rectangular range with the number of rows and columns matching the dimensions of the matrix (e.g., a matrix with 3 rows and 6 columns would have to map to a range in the spreadsheet 3 rows long and 6 columns wide).

If required, you can control the sheet and cell range into which GoldSim exports data dynamically. For example, you could instruct GoldSim to change the location to which it exports data based on the simulation time or the realization.

Read more: Defining Offsets for Inputs and Outputs to a Spreadsheet Element (page 923).

If you press the Import… button from the Import or Export Selection dialog (and the checkbox for using the wizard is cleared), or if you select an existing output in the main Spreadsheet dialog and press the Edit… button, the following dialog is displayed:

This dialog is used to create an output to the Spreadsheet element. An output to a Spreadsheet element is defined by specifying a cell range in the spreadsheet file that you wish to import from the spreadsheet into GoldSim, making the data accessible within GoldSim as an element output.

You define an output as follows:

1. The Name of the output will default to Outputn, where n is an integer. You will use this name to reference the output within GoldSim (as SpreadsheetElementName.OutputName). Hence, you will likely want to change this name from its default. The optional Description appears only in this dialog and is useful for helping to document your spreadsheet link.

2. Press the Type… button to define the attributes of the output. The following dialog will be displayed:
This dialog allows you to define the Order (scalar, vector, matrix) of the output. You can also define the Type. By default, the Type is a Value (conditions are not supported and cannot be selected here). However, you can also select Probability Distribution, which allows you to import a series of input fields that define a distribution.

**Read more:** Importing Stochastic Element Definitions from a Spreadsheet (page 918).

3. You must then specify the units of the date being imported from the spreadsheet. There are two ways to do this. You can simply specify in this dialog what the units are (and this is the default). In this case, the **Units in Spreadsheet** represents the units of the data in the spreadsheet. This will also become the display units for the output after it is imported into GoldSim. As discussed below, alternatively, you can also import the unit string directly from the spreadsheet.

4. You must then select the location in the spreadsheet from which you wish to import the data. You can do this by specifying the Sheet Name and Cell Range directly. Alternatively, you can press the **Location...** button, which opens the spreadsheet and provides a dialog allowing you to directly select the desired Cell Range using your mouse.

5. You can optionally choose to import the units for the data directly from the spreadsheet (rather than specifying the units directly as in step 3 above). In this case, select the **Units** checkbox in the dialog:
output to have different display units than the units in the spreadsheet (although, of course, they must be of the same dimension).

**Note:** The size of the range specified in the spreadsheet must be consistent with Order (scalar, vector, matrix) that you defined for the output. Hence, a scalar output must map to a single cell; a vector output must map to a single column or row range with the number of rows or columns matching the number of items in the vector; and a matrix output must map to a rectangular range with the number of rows and columns matching the dimensions of the matrix (e.g., a matrix with 3 rows and 6 columns would have to map to a range in the spreadsheet 3 rows long and 6 columns wide). When importing a Probability Distribution, the range is a single cell (the first cell in the string of inputs defining the distribution).

**Note:** Spreadsheets cells can be defined as TRUE or FALSE. If you try to import such a cell into GoldSim, a TRUE cell will be imported as the number 1, and a FALSE cell will be imported as 0.

If required, you can control the sheet and cell range from which GoldSim imports data dynamically. For example, you could instruct GoldSim to change the location from which it imports data based on the simulation time or the realization.

**Read more:** Defining Offsets for Inputs and Outputs to a Spreadsheet Element (page 923).

If you would like to edit the location of several inputs (or outputs) simultaneously by shifting them by sheet, row or column, you can do this from the main Spreadsheet element dialog using the Shift… button.

**Read more:** Shifting Ranges for Inputs and Outputs to a Spreadsheet Element (page 925).

**Note:** Prior to running the model, you can view any spreadsheet data that you plan to import by first selecting the Update Spreadsheet Outputs item in the menu displayed via the Options>> button for the Spreadsheet element, and then viewing the outputs of the element in tool-tips within the element’s output port.

**Note:** If GoldSim cannot find the spreadsheet when you try to run the model, the model will still run and GoldSim will log a warning message to the Run Log. If data from the spreadsheet was previously imported (i.e., it had been run previously with the spreadsheet accessible), it will use the data that was imported previously. If data from the spreadsheet was never imported (i.e., it had never been run previously with the spreadsheet accessible), all values will default to 0.

### Importing Stochastic Element Definitions from a Spreadsheet

You can use the Spreadsheet element to import a probability distribution from a spreadsheet. The distribution is defined in the spreadsheet using a number of fields that specify the type of distribution, as well as the arguments to the distribution.
Probability distributions must be specified in the spreadsheet in a very specific format. In particular, the distribution must be defined in a row of adjacent cells in the spreadsheet. The first cell is a code that defines the type of distribution. The subsequent cells (to the right of the code) define the input parameters for the distribution. The table below specifies how each type of distribution must be entered in the spreadsheet.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Type Code</th>
<th>Distribution Parameters</th>
</tr>
</thead>
</table>
| Generalized Beta | BETAGEN    | Cell1: Mean  
                                 | Cell2: SD  
                                 | Cell3: Min  
                                 | Cell4: Max  |
| Beta             | BETASF     | Cell1: Successes  
                                 | Cell2: Failures  |
| BetaPERT         | BETAPERT   | Cell1: Min (0%) or 10%  
                                 | Cell2: Most Likely  
                                 | Cell3: Max (100%) or 90%  
                                 | Cell4: 0 for 0/100; 1 for 10/90  |
| Binomial         | BINOM      | Cell1: Batch Size  
                                 | Cell2: Prob  |
| Cumulative       | CUM        | Cell1: # pairs  
                                 | Cell2: Linear = 0; Log = 1  
                                 | Cell3: Prob1  
                                 | Cell4: Val1  |
| Discrete         | DISCRETE   | Cell1: # pairs  
                                 | Cell2: Prob1  
                                 | Cell3: Val1  
                                 | Cell4: Prob2  
                                 | Cell5: Val2  |
| Exponential      | EXPON      | Cell1: Mean  |
| Extreme Probability | EXTRPROB  | Cell1: Min = 0; Max = 1  
                                 | Cell2: Number samples  |
| Extreme Value    | EXTRVAL    | Cell1: Min = 0; Max = 1  
                                 | Cell2: Location  
                                 | Cell3: Scale  |
| Gamma            | GAMMA      | Cell1: Mean  
                                 | Cell2: SD  
                                 | Cell3: (Min)  
                                 | Cell4: (Max)  |
| Log-Normal       | LOGNORM    | Cell1: True = 0; Geom = 1  
                                 | Cell2: True or Geom Mean  
                                 | Cell3: True or Geom SD  
                                 | Cell4: (Min)  
                                 | Cell5: (Max)  |
| Negative Binomial | NEGBINOM   | Cell1: Number successes  
                                 | Cell2: Prob of success  |
| Normal           | NORMAL     | Cell1: Mean  
                                 | Cell2: SD  
                                 | Cell3: (Min)  
                                 | Cell4: (Max)  |
| Pareto           | PARETO     | Cell1: Shape  
                                 | Cell2: Mode  
                                 | Cell3: (Max)  |
| Pearson Type III | PEARSON    | Cell1: Location  
                                 | Cell2: Scale  
                                 | Cell3: Shape  |
| Poisson          | POISSON    | Cell1: Expected Value  |
| Sampled Results  | SAMPLED    | Cell1: # Results  
                                 | Cell2: Val1  
                                 | Cell3: Val2  
                                 | Cell4: Val3  
                                 | Cell5: Val4  |
| Student’s t      | STUDENTT   | Cell1: Deg of Freedom  |
| Triangular       | TRIANG     | Cell1: Linear = 0; Log = 1  
                                 | Cell2: Min (0%) or 10%  
                                 | Cell3: Most Likely  
                                 | Cell4: Max (100%) or 90%  
                                 | Cell5: 0 for 0/100; 1 for 10/90  |
| Uniform          | UNIFORM    | Cell1: Linear = 0; Log = 1  
                                 | Cell2: Min  
                                 | Cell3: Max  |
| Weibull          | WEIBULL    | Cell1: Min  
                                 | Cell2: Slope  
                                 | Cell3: Mean – Min  
                                 | Cell4: (Max)  |

Table Notes:

- Cumulative and Discrete can have as many Prob/Val pairs as required.
- Sampled Results can have as many Values as required.
Parameters in parentheses are optional.

If Log-Normal distribution is defined using Geometric parameters, the SD is dimensionless.

Shaded parameters are dimensionless; others have same dimension as the distribution.

There is no option to import a Boolean distribution.

To import a distribution from a spreadsheet, you must press the **Type** button in the Import dialog:

After doing so, the following dialog will be displayed. You must select “Probability Distribution” as the Type:

The Order of a Probability Distribution must be Scalar.

When you specify the Location, you must select a single cell: that corresponding to the Type Code for the distribution. GoldSim will automatically read in the appropriate number of additional cells based on the distribution type.

**Note:** Prior to running the model, you can view any spreadsheet data that you plan to import by first selecting the **Update Spreadsheet Outputs** item in the menu displayed via the **Options >>** button for the Spreadsheet element, and then viewing the outputs of the element in tool-tips within the element’s output port.
Note: Care must be taken when importing distributions that represent dates and temperatures. This is because dates and temperatures both have an absolute reference (i.e., 12/31/1899 for dates and absolute zero for temperatures). Differences between dates are expressed in units of time (e.g., days, seconds, etc) and differences between temperatures are expressed in ‘deg’ units (Cdeg, Fdeg). This can lead to errors when importing Stochastics, since GoldSim assumes a single unit for the distribution parameters, while some types of distributions would require two different types of units. For example, a Normal distribution representing a temperature would require the Mean to be specified in absolute units (e.g., C) and the Standard Deviation to be specified in difference units (e.g., Cdeg). Similarly, a Normal distribution representing a date would require the Mean to be specified in absolute units (e.g., as a date) and the Standard Deviation to be specified in difference units (e.g., days). If faced with this problem, there are several approaches for addressing this: 1) Use a distribution type where all parameters have the same units (e.g., triangular, uniform, cumulative); or 2) Specify and import the distributions as dimensionless and then apply a unit to the sampled value using an Expression element. This latter option would work for temperatures, but not dates, since a date cannot be specified with dimensionless values. It could, however, be imported as a Julian date, in which case you could use day units (or some other time unit). Alternatively, the mean date could be defined as a Data element and an imported distribution with time units could be used to model a ‘lag’ (positive or negative) from the fixed date. An Expression element could be used to add the sampled lag value to the fixed (mean) date.

Read more: Dealing with Temperature Units (page 92).

When a distribution is imported in this way, the output type on the spreadsheet element is not a value; rather, it is a complex output referred to as a Distribution output. Distribution outputs are complex outputs that represent all the statistical information necessary to define a probability distribution and only be used in several specialized locations.

To import this information into a Stochastic element, you must then use this Distribution output as the definition for a Stochastic element specified as an Externally-Defined distribution.

Read more: Externally-Defined Distribution (page 167).

If you press the Import... button from the Import or Export Selection dialog, and the checkbox for using the wizard is checked, the Spreadsheet Wizard will be displayed:
Using External Application Elements

Chapter 10: Advanced Modeling Concepts

GoldSim User’s Guide

This wizard is used to create an output to the Spreadsheet element. An output to a Spreadsheet element is defined by specifying a cell range in the spreadsheet file that you wish to import from the spreadsheet into GoldSim, making the data accessible within GoldSim as an element output. Until you become comfortable creating inputs and outputs for Spreadsheet elements, it is recommended that you use the wizard.

The wizard has three pages:

1. On the first page of the wizard, you select the location in the spreadsheet from which you wish to import the data. You can do this specifying the Sheet Name and Cell Range directly. Alternatively, you can press the Location… button, which opens the spreadsheet and provides a dialog allowing you to directly select the desired cell range using your mouse.

Note: You cannot use the wizard to import a probability distribution from a spreadsheet. To do this, you must define the Import manually.

Read more: Importing Stochastic Element Definitions from a Spreadsheet (page 918).

2. On the second page, you must specify the Units in Spreadsheet. These represent the units of the data in the spreadsheet. This will also become the display units for the output after it is imported into GoldSim. If the range you have selected in Step 1 is more than a single cell (indicating that the data will be imported into GoldSim in the form of a vector or matrix), you will be prompted for the Array Labels for the vector or matrix.

Note: You cannot use the wizard to import the units for the data from a spreadsheet. To do this, you must define the Import manually.

Read more: Spreadsheet Element Outputs: Importing Data from the Spreadsheet (page 916).
3. On the last page of the wizard, you specify the Name of the output. It will default to Output, where n is an integer. You will use this name to reference the output within GoldSim (as SpreadsheetElementName.OutputName). Hence, you will likely want to change this name from its default.

**Note:** If you specify a spreadsheet range that is more than one cell, the size of the range specified in Step 1 must be consistent with the size of the Array Label set(s) specified in Step 2. Hence, a vector output must map to a single column or row range with the number of rows or columns matching the number of items in the vector; and a matrix output must map to a rectangular range with the number of rows and columns matching the dimensions of the matrix (e.g., a matrix with 3 rows and 6 columns would have to map to a range in the spreadsheet 3 rows long and 6 columns wide).

If required, you can control the sheet and cell range into which GoldSim exports data dynamically. For example, you could instruct GoldSim to change the location to which it exports data based on the simulation time or the realization.

**Read more:** Defining Offsets for Inputs and Outputs to a Spreadsheet Element (page 923).

**Note:** Prior to running the model, you can view any spreadsheet data that you plan to import by first selecting the Update Spreadsheet Outputs item in the menu displayed via the Options>> button for the Spreadsheet element, and then viewing the outputs of the element in tool-tips within the element’s output port.

**Defining Offsets for Inputs and Outputs to a Spreadsheet Element**

In some cases, you may want to specify the location in the spreadsheet file to which you are linking in terms of an offset from a defined location, in which the offset is defined by model variables (such as Time or Realization).

For example, you could instruct GoldSim to change the location to which it exports data based on the realization number, with results from each realization being exported to a different column in the spreadsheet file.

You do this by selecting the Offset... button when you are selecting a spreadsheet location while defining inputs or outputs to a Spreadsheet file:

![Defining Offsets dialog](image)

When you press this button, the following dialog is displayed:
You select whether you want to offset by Sheet, Column, and/or Row by clicking the appropriate checkboxes. For example, if you were exporting a scalar GoldSim output to the spreadsheet file, and wanted to export the value to Sheet1!A1 for realization 1, Sheet1!B1 for realization 2, and so on, you would simply do the following:

1. Specify Sheet1!A1 for the Location in Spreadsheet
2. Define the Column offset as follows:

In this case, the offset would be 0 for Realization 1 (and hence the value would be exported to Sheet1!A1), it would be 1 for Realization 2 (and hence the value would be exported to Sheet1!B1), and so on.

**Note:** The offsets can be any expression with links to other GoldSim outputs, but should be specified as dimensionless, integer values. GoldSim will not accept expressions in these fields that have dimensions. Offsets defined as real numbers are truncated.

**Note:** If as a result of an offset, you try to access a cell that is invalid (e.g., if you specify a negative offset from the first column), GoldSim will display a fatal error message at runtime.

If you wish to use a particular unit of time to define an offset, you must cast the output to a dimensionless number. For example, Time|day| will represent the number of days (which could be fractional, and hence will be truncated). Note that Run Properties like Day and Year are, by definition, dimensionless.

**Read more:** Unit Casting (page 93); Understanding and Referencing Run Properties (page 486).

When you close the Offset dialog and have a defined offset, this is indicated in the Spreadsheet Location (Offset… is checked) and in the list of inputs and outputs (any offset ranges are indicated as such).
Warning: In order to properly use offsets, you must have a good understanding of the conditions and settings that control when GoldSim exchanges data with the spreadsheet file.

Read more: Controlling When GoldSim Exchanges Data with the Spreadsheet File (page 927).

An example model which includes an illustration of the use of offsets for a Spreadsheet element (Spreadsheet.gsm) can be found in the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

In some cases, you may want to edit the location of several inputs (or outputs) simultaneously by shifting them by sheet, row or column.

This could happen, for example, if you have multiple Spreadsheet elements linked to a single spreadsheet with multiple sheets. You may want each Spreadsheet element to link to a different sheet of the file, but otherwise, you would like all of the row and columns in the links to be identical.

GoldSim provides a very convenient way to shift multiple inputs or outputs by sheet, row, and/or column.

You do this by selecting the items that you want to shift, and pressing the Shift... button from the main Spreadsheet element dialog:

Multiple contiguous rows in this dialog can be selected by holding down the Shift key while selecting with the mouse. Multiple rows that are not contiguous can be selected by holding down the Ctrl key while selecting with the mouse. All rows can be selected by placing the cursor in any row and pressing Ctrl-A.
When you select items and press the **Shift**… button, the Spreadsheet element dialog expands to show the following section:

The **Up** and **Down** keys shift the selected items by row. The **Left** and **Right** buttons shift the selected items by column.

**Note:** If you try to shift beyond the valid range for any of the selected items (e.g., if you try to shift left and one of the items is defined as being in column A) GoldSim will not shift any of the items.

The **Change Sheet**… button displays a dialog for selecting a new sheet to apply to all selected items:

The drop-list includes all worksheets present in the Excel file. By default, no new sheet is selected. If you press OK without selecting a sheet, no changes are made. If you select a sheet and press OK, all selected items will be updated (they will point to that sheet).

**Note:** If you are importing both the data and the units for the data from a spreadsheet, when you shift the data, GoldSim will also shift the reference to the unit if a) you are shifting rows, and the data and the units are in the same row; b) you are shifting columns, and the data and the units are in the same column; or c) you are shifting sheets (by definition, the units are always in the same sheet as the data).

**Locking onto a Spreadsheet File**

GoldSim allows you to “lock onto” the spreadsheet file that is being referenced. When you lock onto a file (by selecting the **Lock onto selected file** item in the menu displayed via the **Options**>> button), the following additional information regarding the referenced file is saved with the element:

- File and path name;
• Date the file was created;
• Date the file was last modified;
• File size; and
• CRC signature.

Once you have locked onto a file, the CRC signature is displayed in a tool-tip when you hold your cursor over the filename in the dialog.

The CRC signature is an alphanumeric code that can be used to uniquely identify whether the file contents have changed. When you run a model that has locked onto a file, GoldSim compares the CRC signature of the file with the original signature that was stored. If these are not identical (indicating that the file has been changed), GoldSim displays an error message and will not run the model.

You can unlock a file by clearing the Lock onto selected file item. Note that if versioning is enabled, whenever a file is locked onto or unlocked, this information is logged with the version.

**Read more:** Tracking Model Changes (page 1023).

**Note:** Saving changes to a spreadsheet file and file locking are mutually exclusive. If the Save MS-Excel file after simulation item is selected in the Options>> menu when you try to lock onto a file, GoldSim will warn you that locking onto the file will automatically disable saving changes to the MS-Excel file. When you lock onto a file, the Save MS-Excel file after simulation item is no longer available in the Options>> menu.

**Note:** The act of opening a workbook in Excel automatically modifies the file, even if no changes are made. In order to allow spreadsheet files to be viewed without changing the CRC signature, GoldSim automatically makes the file read-only when it is locked onto. Note that if it is subsequently unlocked, it remains read only, and you will need to change this manually if you wish to edit the file.

**Read more:** How Spreadsheet Files are Affected by GoldSim (page 929).

Whenever a Spreadsheet element is used in a model, GoldSim uses the following information to determine when during the simulation it should import data from or export data to the spreadsheet file:

- Is the Recalculate in Excel during simulation setting (accessed via the Options>> button) On or Off? This defaults to being On (yes) when a new Spreadsheet element is created.
- Are you importing data from the spreadsheet, exporting data to the spreadsheet, or both?
- Are offsets defined, and if so, has an offset value changed?
- If you are exporting data to the spreadsheet, have any of the GoldSim outputs being exported changed?

The following table summarizes how this information is used to determine when to exchange data with the spreadsheet file:
### Using External Application Elements

#### Chapter 10: Advanced Modeling Concepts

GoldSim User’s Guide

<table>
<thead>
<tr>
<th>Type of Exchange Specified in Element</th>
<th>Recalculate Setting</th>
<th>Offsets defined</th>
<th>When does GoldSim exchange data with spreadsheet file?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Export only</td>
<td>Not available</td>
<td>No</td>
<td>Export all data at end of simulation.</td>
</tr>
<tr>
<td></td>
<td>Not available</td>
<td>Yes</td>
<td>Export data from just before its offset changes.*</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Export all data at end of simulation.</td>
</tr>
<tr>
<td>Import only</td>
<td>Not available</td>
<td>No</td>
<td>Import all data at beginning of simulation.</td>
</tr>
<tr>
<td></td>
<td>Not available</td>
<td>Yes</td>
<td>Import all data at beginning of simulation. Import data when its offset changes</td>
</tr>
<tr>
<td>Export and Import</td>
<td>Off</td>
<td>No</td>
<td>Export all data at end of simulation. Import all data at beginning of simulation.</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>No</td>
<td>Export and Import all data at beginning of simulation. Export and Import all data whenever an Export variable changes. Export all data at end of simulation.</td>
</tr>
<tr>
<td></td>
<td>Off</td>
<td>Yes</td>
<td>Export and Import all data at beginning of simulation. Export data from just before its offset changes.* Export all data at end of simulation. Import data when its offset changes</td>
</tr>
<tr>
<td></td>
<td>On</td>
<td>Yes</td>
<td>Export and Import all data at beginning of simulation. Export and Import all data whenever an Export variable changes. Export data from just before its offset changes.* Export all data at end of simulation. Import data when its offset changes</td>
</tr>
</tbody>
</table>

*The value that is exported when an export offset changes is the value from the update immediately before the offset changed. Hence, if the offset was “Realization”, when Realization changes from 1 to 2, GoldSim would export the value from the end of Realization 1 using an offset of 1. Similarly, if the offset was “Month”, when Month changes from 1 to 2, GoldSim would export the value from the end of Month 1 using an offset of 1.*

Several points about spreadsheet recalculations are worth noting:

- Regardless of the **Recalculate in Excel during simulation** setting, by default, Excel will recalculate whenever cell values change. As pointed out below, however, GoldSim will only retrieve the updated data if **Recalculate in Excel during simulation** is selected. You can change this setting in Excel (so it does not automatically recalculate) in the Calculation tab of Excel’s Options dialog.

- The **Recalculate in Excel during simulation** setting does two things: 1) it forces Excel to recalculate, and 2) it retrieves the recalculated data.
back into GoldSim. Hence, if the **Recalculate in Excel during simulation** setting is off, GoldSim will *not retrieve updated data* from the spreadsheet, even if some cells have changed (and Excel automatically recalculated).

**Note:** When you use a Spreadsheet element, there may be a delay of several seconds at the start of a simulation as Excel is loaded into memory. Once it is loaded, however, the simulation should progress rapidly.

**Note:** If you are only importing data from a spreadsheet *and* you have not specified any dynamic offsets *and* you have already imported the data previously, if GoldSim cannot find the spreadsheet when you try to run the model it will use the data that was imported previously and will log a warning message to the Run Log.

**Warning:** You should never try to open a spreadsheet that is linked to GoldSim while a simulation is running. In addition, prior to running a GoldSim model that is linked to a spreadsheet, all referenced spreadsheets should be closed. If you try to interact with a spreadsheet that is being used by GoldSim during a simulation, all interaction with the spreadsheet is blocked by GoldSim.

If you are importing data from a spreadsheet, you can tell GoldSim to import the data while in Edit Mode by selecting **Update Spreadsheet Outputs** via the **Options>>** menu button. When you do this, GoldSim will import the data currently existing in the spreadsheet. If the information being imported is impacted by data that is being exported to the spreadsheet (and that data can be computed in Edit Mode), GoldSim will export the data to the spreadsheet, trigger a recalculation, and import the data from the spreadsheet to GoldSim, where it can be viewed as an output of the spreadsheet element.

By default, GoldSim’s interactions with the Excel spreadsheet only affect a copy of the spreadsheet temporarily stored in memory, and do not affect the data permanently stored in your spreadsheet file. That is, following your simulation, the spreadsheet file will be unchanged (even if during the simulation GoldSim has sent data to cells in the spreadsheet).

In some cases, however, you may want to save the changes you make to a spreadsheet (i.e., when GoldSim exports data to the spreadsheet). To do so, select the **Save MS-Excel file after simulation** item in the **Options>>** menu (which by default is cleared).

**Note:** Unless you use offsets that vary with time or realization, if you access the spreadsheet multiple times (at multiple timesteps or for multiple realizations), only the changes made to the spreadsheet the final time you access it will be saved (all the previous modifications will be overwritten each time that you access it).

**Read more:** [Defining Offsets for Inputs and Outputs to a Spreadsheet Element](page 923).
Note: Saving changes to a spreadsheet file and file locking are mutually exclusive. If the **Save MS-Excel file after simulation** item is selected in the Options menu when you try to lock onto a file, GoldSim will warn you that locking onto the file will automatically disable saving changes to the MS-Excel file. When you lock onto a file, the **Save MS-Excel file after simulation** item is no longer available in the Options menu.

Read more: [Locking onto a Spreadsheet File](#) (page 926).

Warning: You should always close any spreadsheets that are referenced by GoldSim prior to running your GoldSim model. In fact, in most cases, GoldSim will warn you if it tries to interact with (in particular, write to) a spreadsheet that is already open.

### Exchanging Date Information with a Spreadsheet

In some cases, you may wish to exchange date or time information with a spreadsheet. For example, you may wish to send the current date in the GoldSim simulation to a cell formatted as a date in a spreadsheet, or retrieve a date from a spreadsheet into GoldSim. In order to do so, you must understand how Microsoft Excel (and GoldSim) deal with dates.

Within Excel, a date is stored internally as the number of days (which can be fractional) since December 31, 1899 00:00:00. Hence, if you were to export the number 4.25 into a spreadsheet cell formatted as a date, it would be interpreted as January 4, 1900 at 6:00:00 AM. However, Excel mistakenly adds an extra day into its calendar that did not actually exist (February 29, 1900). As a result, the *effective* Julian date reference for all times after March 1, 1900 in Excel is actually December 30, 1899 00:00:00.

Within GoldSim, the Run Property DateTime represents the elapsed time since the reference date December 30, 1899 00:00:00. Hence, for all dates after March 1, 1900, Excel and GoldSim have the same effective Julian date reference.

Given this information, the following rules should be followed for exchanging date information between GoldSim and a spreadsheet:

- When exporting a date/time from GoldSim to a spreadsheet cell, you should select Days for the **Units in Spreadsheet**. If the spreadsheet cell is formatted as a date, it will display as a date; otherwise, it will display the number of days from December 30, 1899 00:00:00 to the specified date that was exported.

- When importing a date from a spreadsheet cell (i.e., a cell formatted in the spreadsheet as a date), you should select Date or Datetime for the **Units in Spreadsheet**. Within GoldSim, the output from the Spreadsheet element would then be stored as the number of days from December 30, 1899 00:00:00 to the particular date that was imported from the spreadsheet. If running a Calendar Time simulation, the output would be displayed as a date when viewing it (e.g., using a tool-tip) in GoldSim.

Read more: [Referencing Dates in Expressions](#) (page 131); [Setting the Basic Time Options](#) (page 453).

### Saving Spreadsheet Element Outputs

You can save results for a Spreadsheet element by clicking **Final Values** and/or **Time Histories**. Checking one of these causes all output arguments to be saved.
If the element has multiple outputs, and you wish to save only one or two of these as results, you can use the context menu of the output in the browser (accessed via a right-click on an output) to turn on or off a particular output. In this case, the checkbox in the dialog would then become a box instead of a check (indicating that only some of the results will be saved). Note that in order to see outputs in the browser you must first ensure that Show element subitems is selected by right-clicking anywhere in the browser.

Note: A Spreadsheet element does not have a primary output.

Viewing a Spreadsheet Element in the Browser

The browser view of a Spreadsheet element is shown below:

As can be seen, the browser view shows an item for each specified input to the spreadsheet, and an item for each specified output.

Note: Element inputs and outputs are only shown in the browser if you choose to Show element subitems (accessed via the browser context menu by right-clicking in the browser).

External (DLL) Elements

In some situations, you may wish to define a complex function which cannot be readily implemented using the expression editing features supplied by GoldSim. For example, calculation of an output may require very complex logic which would be cumbersome to represent using a Selector element, or it may require a numerical solution technique (e.g., iteration).

GoldSim provides two separate ways to deal with such situations:

- You can develop separate program modules (written in C, C++, FORTRAN or other compatible programming languages) which can then be directly coupled with the main GoldSim algorithms. These user-defined modules are referred to here as external functions, and the elements through which they are coupled to GoldSim are called External (DLL) elements.
- You can specify a sequence of statements directly within the GoldSim interface using a Script element.

Read more: Script Elements (page 860).

In general, using Script elements is preferred, as this approach has the key advantages that 1) it does not require use of a separate programming language and interface; and 2) it is much more transparent (all of the “code” can be seen directly in GoldSim). However, in some cases (e.g., linking complex existing programs directly into GoldSim), Script elements cannot be used and External (DLL) elements are the only option.

The modules are linked into GoldSim as DLLs (Dynamic Link Libraries) at run time. Integrating these external modules into GoldSim requires that you develop...
a "wrapper" (or "shell") around your existing function, and compile it into a DLL. In most cases, this will require only a limited number of programming modifications. GoldSim supports both 32-bit and 64-bit DLLs.

An example model which uses an External (DLL) element (External.gsm) can be found in the General Examples/External folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu). This folder also includes the corresponding DLL, as well as the source code (in C++ and Fortran) for the example.

The steps involved in creating and using an External element are as follows:

1. Code the external function to be used by the External element in your model;
2. Compile the external function into a DLL (multiple functions can be included in the same DLL);
3. Create an External element within GoldSim and specify the DLL and the specific function within the DLL to be used;
4. Specify the input and output arguments that are to be sent to and received from the function.

The details of the first two steps are discussed in Appendix C. The remaining two steps are discussed below.

Read more: Appendix C: Implementing External (DLL) Elements (page 1095).

The editing dialog for an External element is shown below:

In order to use an External element, you must first specify the **DLL Path** and **Function Name** within the DLL that the element will utilize:

**DLL Path**: This is the name of the dynamic link library containing the external function. You can press the Browse button (the folder to the right of the DLL Path field) to search for the file. In general, you should specify the full path to the DLL. (If you specify just the filename, it will look for
the DLL only in the working directory containing the GoldSim model file). Note that a single DLL can contain multiple external functions.

**Note:** If the DLL references any external files and it is run in a separate process, it will look for those files in the folder where the DLL resides. If it is not run in a separate process, it will look for the files in the folder where the model file resides.

**Function Name:** This is the specific name of the external function in the DLL. This name is case-sensitive and must exactly match the name of the function in the source code for the external function.

There are also several options that control how the DLL is called by GoldSim:

- **Unload DLL after each use:** If this box is checked, GoldSim will unload the DLL (and continue the simulation). This is useful when running very large model files (in which the DLL only needs to be called infrequently). If the DLL is subsequently called again, GoldSim will automatically reload it.

- **Run Cleanup after each realization:** If this box is checked, GoldSim will call the DLL with a cleanup instruction (99) at the end of each realization.

  Read more: [Appendix C: Implementing External (DLL) Elements](page 1095).

- **Lock onto this file:** If this box is checked, GoldSim regards various properties of the file to determine whether the file contents have changed.

  Read more: [Locking onto a DLL File](page 936).

- **Run in separate process:** If this box is checked, GoldSim executes the DLL outside of the GoldSim process space. This can be useful for DLLs that have large memory requirements.

  Read more: [Running the DLL in a Separate Process](page 937).

**Note:** You can use GoldSim's File element to automatically copy the DLL file from some location (typically on a network) to your machine. Because GoldSim records this action in the Run Log, it can provide an "audit trail" to ensure that the proper version of an external file (such as a DLL) was used in a particular simulation.

Read more: [File Elements](page 941).

If you check the **Record CPU times in the run log** box, if the element uses more than 1 CPU seconds, a message will be written to the GoldSim run log identifying the element’s name, type (i.e., External), the number of times it was updated, and the total CPU time used.

Read more: [The Run Log](page 548).

You define the input and output arguments for the External element via the **Interface** tab:
The top part of this dialog is used to define the input arguments. The bottom portion is used to define the output arguments.

In order to add an input argument, press the Add button in the Input Interface portion of the dialog. When you do this, GoldSim displays the Insert Link dialog, allowing you to select an existing output to add to the interface. After selecting a output, the following dialog is displayed:

The **Definition** displays the output that was selected. The **Name (ID)** defaults to the output name. The **Name (ID)** will appear on the input interface for the External element (accessed via the input port). The **Description** is optional and simply provides additional information regarding the input argument (it defaults to the element description for the output that was selected).

Before sending the input to the DLL, GoldSim converts the value to the **External Units** (the units themselves are not sent to the DLL, just the value). The **External Units** and **Type...** (e.g., value/condition, scalar/vector/matrix) for the input argument are automatically determined by the output that was selected and linked to. Moreover, the **External Units** themselves default to the Display Units for the linked output. Of course, the DLL may require that the value be sent to it assuming different units. Hence, you can edit the **External Units** specified (but, of course, these must have dimensions consistent with the **Definition**).
**Note:** If you send an output whose **Type** is a Condition to an external function, GoldSim will automatically convert it to either a 0 (for False) or 1 (for True) prior to sending the argument to the DLL.

**Note:** If you pressed **Cancel** (rather than selecting an output) when adding the input argument, the **Definition** (and hence **External Units** and **Type**) will be blank. You could enter any expression into the **Definition** field (including constants). The **External Units** and **Type** would need to be specified in a manner that was consistent with the **Definition**.

**Note:** In addition to reading in Values or Conditions, an External function can also read in (and subsequently output) a Time Series Definition.

**Read more:** [Using an External Element to Read and/or Output Time Series](page 939).

In order to add an output argument, press the Add button in the Output Interface portion of the dialog. When you do this, GoldSim displays the following dialog:

![Add button](image)

The **Name (ID)** is the name by which the output variable can be referenced in the model. For example, if the name of the External element was X, and the name of the output was Y, the output could be referenced as X.Y. This ID has the same restrictions as an element ID.

The **Description** is an optional description of the output. The **External Units** represent the units in which the External function returns the value. The specified units also become the display units for the output. The **Type**... button is used to access a dialog for specifying the data type (e.g., value/condition, scalar/vector/matrix).

**Note:** If you add an output whose **Type** is a Condition to the output interface, GoldSim will treat any non-zero number as True, and zero as False.

In addition to defining the output as a Value or a Condition, the output can also represent either a Lookup Table or a Time Series:
There is no limit to the number of input and output arguments that can be added. After adding the input and output arguments, the Interface tab displays them as follows:

The input arguments and output arguments are transferred between GoldSim and the external function in exactly the order in which they are listed on the Interface tab.

Buttons for deleting, editing and moving the input and output arguments up and down in the list are available directly below the Add buttons.

**Locking onto a DLL File**

GoldSim allows you to “lock onto” the DLL file that is being referenced. When you lock onto a file (by checking the **Lock onto this file** option in the External element dialog), the following additional information regarding the referenced file is saved with the element:

- File and path name;
- Date the file was created;
- Date the file was last modified;
- File size; and
Using External Application Elements

• CRC signature.

Once you have locked onto a file, this information is displayed in a tool-tip when you hold your cursor over the filename in the dialog.

The CRC signature is an alphanumeric code that can be used to uniquely identify whether the file contents have changed. When you run a model that has locked onto a file, GoldSim compares the CRC signature of the file with the original signature that was stored. If these are not identical (indicating that the file has been changed), GoldSim displays an error message and will not run the model.

You can unlock a file by clearing the Lock onto this file checkbox. Note that if versioning is enabled, whenever a file is locked onto or unlocked, this information is logged with the version.

Read more: Tracking Model Changes (page 1023).

By default, GoldSim loads DLLs used by External elements into the GoldSim process space. GoldSim and the all external DLLs therefore share the application’s address space, which is usually limited to 2 GB on Windows operating systems. In some case, a single DLL itself may have large memory requirements (e.g., if it loaded large sets of data from files or databases), or it could statically or dynamically allocate large blocks of memory. If the combined memory requirements of GoldSim (the executable), the GoldSim model (including all model properties, buffers and result values) and all DLLs exceeds 2 GB, GoldSim will be unable to carry out the simulation. The user will be forced to preserve memory by saving fewer histories or time steps, or by turning off saving of results for certain elements.

To circumvent this limitation, GoldSim provides a feature that allows external DLLs to be executed outside of the GoldSim process space. If Run in separate process is checked, GoldSim does not load the DLL into its process space. Instead, it launches a separate small DLL server with its own private process space. The DLL server works like a middleman who dispatches information between GoldSim and the external DLL. Each DLL that runs in a separate process has its own DLL server. Therefore each DLL has access to a private 2 GB process space, which is not shared with any other DLL or application.

Note: If the DLL references any external files and it is run in a separate process, it will look for those files in the folder where the DLL resides. If it is not run in a separate process, it will look for the files in the folder where the model file resides.

Note: 64-bit DLLs must be run in a separate process.

Using an External Element to Define Lookup Tables

In some cases, you may wish an external program to directly generate a look-up table that you can use to dynamically define a Lookup Table element.

Read more: Using Lookup Table Elements (page 294).

To facilitate this, GoldSim allows you to define the table using an External element. The entire table then becomes an output of the External element, which can subsequently be linked to a Lookup Table element.
In order to specify that an output of the External element is a table, you must indicate that the Data Type of the output is a 1D, 2D, or 3D Table Definition (when you define the Type... for an output argument):

As seen above, in addition to defining an output of an External element as a Value or a Condition, you can also define an output whose data type is a 1D, 2D, or 3D Table Definition. These three special data types can only be produced by an External element and outputs of these types can only be used in special locations within GoldSim (i.e., an input defining a Table element's data).

Since the table itself is being defined by an external function (and will subsequently be linked to a Lookup Table element), it is necessary to specify the units for the numbers representing the table that are being passed into GoldSim by the external function.

As a result, when you add an output argument and define it as a Lookup Table, the dialog for defining the output differs depending on whether the data type is a 1D, 2D, or 3D Table Definition. In all cases, the External Units represent the units for the independent variable. For a 1D Table Definition, you must also define the units for a single independent (Row) variable. For a 2D Table Definition, you must define the units for the two independent (Row and Column) variables. For a 3D Table Definition, you must define the units for the three independent (Row, Column and Layer) variables. Here is an example of what the output argument dialog looks like for a 2D Table Definition:

You link a Lookup Table to an external function by selecting “External DLL” from the Data Source field at the bottom of the Lookup Table dialog:

When you do so, a new tab (External DLL) is added to the dialog. You use this tab to specify the name of the External element output that defines the table.
**Note:** In order to link a 1D, 2D, or 3D Table Definition output from an External element to a Table element, the dimensions of the External Units specified in the dialog for the output must be consistent with the Result Units and the independent variable units specified when defining the Lookup Table element.

**Note:** After you have defined a Lookup Table element externally and run the model (so that the DLL has loaded the table into the Lookup Table element), you can view the table definition created by the external function by pressing View Data…. In addition, if you have run the model and you subsequently "uncouple" the External element from the Lookup Table element (by selecting “None” from the Data Source drop-list in the Lookup Table dialog), the data in the table will become accessible (by pressing Edit Table…).

The specific format in which an external function passes table definitions to GoldSim is discussed in Appendix C.

In some cases, you may wish an external program to directly read, manipulate, and/or generate a Time Series Definition. A Time Series Definition output produced by such an external function can be used to dynamically define a Time Series element.

**Read more:** Referencing a Time Series Definition Output (page 225).

To facilitate this, GoldSim allows you to input and/or output a Time Series Definition using an External element.

In order to specify that an input argument of the External element is a Time Series Definition, you must select a Time Series Definition output when you select the output to link to. After doing so, the input argument dialog will look like this:

![Edit External Input](image)

Note that in addition to listing the Definition and External Units, it is also necessary to specify what the time series Data Represents. The External Units and Data Represents field must be consistent with the actual Time Series Definition that is referenced (and automatically default to the correct values when the link is created).

**Read more:** Specifying What the Input to a Time Series Represents (page 190).

In order to specify that an output argument of the External element is a Time Series Definition, you must indicate that the Data Type of the output is a Time Series Definition (when you define the Type… for an output argument):
As seen above, in addition to defining an output of an External element as a Value, Condition or Table, you can also define an output whose data type is a Time Series Definition. Time History Definition outputs are complex outputs that represent all the information necessary to define a time series.

If you select Time Series Definition, the following dialog is displayed for specifying the attributes of the output:

Note that in addition to defining the External Units, you must also specify what the time series Data Represents.

Read more: Specifying What the Input to a Time Series Represents (page 190).

You link a Time Series Definition produced by an External element to a Time Series element by selecting "Linked to external Time Series Definition" from the Data Source field in the Data Definition section of the Time Series element dialog:

When you do so, a new tab (Linked) is added to the dialog. You use this tab to specify the name of the External element output that defines the time series.

Note: If using an External element to output a Time Series Definition, the DLL can only be called at etime = 0. Because a Time Series must be defined at the beginning of the simulation, if you try to redefine the time series in the middle of the simulation, GoldSim will issue a Fatal Error.

The specific format in which an external function reads and outputs Time Series Definitions is discussed in Appendix C.

When is the External Element Called?

The external function is called at the following times:

- At the beginning of the simulation (Etime = 0);
Every GoldSim timestep (if one of the inputs to the element has changed); and

- At any internal step in which an Event is triggered (if one of the inputs to the element has changed).

**Read more:** [How GoldSim Inserts Events into a Simulation](page 417).

It is likely that you may want to treat each of these calls in a different way. For example, you may wish the external function to ignore the call at the beginning of the simulation, or you may want it to carry out some special initialization routines at that time.

You may also want to treat a call in the middle of a timestep (due to an Event) differently from a call at the end of a timestep.

In order to communicate this kind of information to the external function, you would need to send Etime (the current elapsed time) and perhaps Timestep_Length (the length of the current timestep) to the external function as optional input arguments. (Both of these are Run Properties).

**Read more:** [Understanding and Referencing Run Properties](page 486).

You can save results for an External element by clicking Final Values and/or Time Histories. Checking one of these causes all output arguments to be saved. If the element has multiple outputs, and you wish to save only one or two of these as results, you can use the context menu of the output (accessed via a right-click) in the browser to turn a particular output on or off. In this case, the checkbox in the dialog will become gray (indicating that some of the results will be saved). The display units for these outputs are defined when the output is created.

**Note:** An External element does not have a primary output.

The browser view of an External element is shown below:

**Saving External Element Outputs**

The browser view shows an item for each specified input argument, and an item for each specified output argument. The names of the inputs and outputs are determined by the **Name (ID)** specified when you added the argument to the interface.

**Viewing an External Element in the Browser**

The File element is a support element in GoldSim. It has no inputs or outputs and carries out no calculations. Its purpose is to simply control copies of auxiliary files that may be required by External elements and Spreadsheet elements.
File elements have two uses:

- To ensure that all necessary support files are passed to Slaves when using the GoldSim Distributed Processing Module; and
- To ensure that support files which are stored on a network are accessed for use in a simulation, and to provide an “audit trail” of the file transfer.

In the latter case, the File Element utilizes a specifically formatted database to identify a "source" file, typically on a network, and then copy the target file onto the local computer. This file is then available for access by GoldSim elements.

The dialog for a File element looks like this:

The most common use of a File element is to ensure all necessary support files are passed to Slaves when using the GoldSim Distributed Processing Module.

The Distributed Processing Module allows you to use multiple computers connected over a network to share the computational burden of a Monte Carlo simulation. This is accomplished by having a Master (server) GoldSim executable connect to multiple Slave (client) GoldSim executables.

The first step in such a simulation is for the Master to transfer the necessary files (the model files and any support files) to the Slaves.

If the model uses any External or Spreadsheet elements, the Master automatically transfers the DLLs and spreadsheet file, respectively, to the Slaves. If, however, these elements reference any other files (e.g., if the DLL reads in a separate data file the first time it is called), GoldSim has no way of knowing this, and could not send the required files to the Slaves.

The File element can be used to overcome this problem. Any file referenced by a File element is automatically sent to Slaves in a distributed processing simulation.

Hence, when using the Distributed Processing Module with external function elements that reference other files, a File element should be created for each file.

The file name should be entered in the Local file name field of the File element, and the Database checkbox should be cleared. If no path is provided, the file should be placed in the same directory as the model file, and if the file cannot be found locally, the Status area displays “Local file missing”, and the simulation is not allowed to proceed.
Using File Elements to Access a File on a Network

A File element can be used to ensure that support files which are stored on a network are accessed for use in a simulation, and to provide an “audit trail” of the file transfer.

When used in this way, the Element ID for the File element is used as a key to access a record in a specifically formatted database referred to as an Extended GoldSim database. This record provides a network path-name for the source file, and also provides an integrity-checking code for the file (to ensure that the file was copied with no errors).

Read more: Downloading from an Extended GoldSim Database (page 1038).

To use a File element in this way, the Database box must be checked, and you must enter the data source name associated with an Extended GoldSim database. Before using GoldSim’s database features you must first define all available databases on your computer using Control Panel’s 32-bit ODBC Data Sources option. This allows you to define a name for each data source, and link it to a specific database file.

Read more: Adding Data Sources to Your Computer (page 1033).

When the Download Now button is pressed, or when a global database download is carried out, GoldSim locates the source file and makes a copy of it. The Local file name input field defines the local destination for the copy of the source file. If the local file already exists, it is automatically overwritten.

After copying, the CRC signature of the downloaded file is compared to that of the file entered in the database to ensure the integrity of the data transfer. (The CRC signature is an alphanumeric code that can be used to uniquely identify whether the file’s contents have changed.) If the codes are different, the download is treated as having failed.

As is the case for other Extended GoldSim database records, you can optionally Select most recent value no later than, in which case GoldSim will interrogate the database for the most current version of the source file dated on or prior to that date. If this field is left blank, GoldSim will use the default effective date, as specified in the GS_Parameter table. Details in the format of the Extended GoldSim database are provided in Appendix E.

Locking onto a File

GoldSim allows you to “lock onto” the file that is being referenced by a file element. When you lock onto a file (by checking the Lock onto this file option in the File element dialog), the following additional information regarding the referenced file is saved with the element:

- File and path name;
- Date the file was created;
- Date the file was last modified;
- File size; and
- CRC signature.

Once you have locked onto a file, this information is displayed in a tool-tip when you hold your cursor over the filename in the dialog.
Localizing Containers

The CRC signature is an alphanumeric code that can be used to uniquely identify whether the file contents have changed. When you run a model that has locked onto a file, GoldSim compares the CRC signature of the file with the original signature that was stored. If these are not identical (indicating that the file has been changed), GoldSim displays an error message and will not run the model.

You can unlock a file by clearing the Lock onto this file checkbox. Note that if versioning is enabled, whenever a file is locked onto or unlocked, this information is logged with the version.

Read more: Tracking Model Changes (page 1023).

Localizing Containers

In most models, element names are forced to be unique. In some cases, however, you may want to have more than one element with the same name. This requirement usually arises when you have a number of similar subsystems in your model.

For example, suppose that you have developed a subsystem (within a Container) that calculates the balance in a bank account as a function of the previous balance, interest rate, deposits and withdrawals. Each of these items would be represented by its own element in the Container. After creating these elements, you decide you would like to track ten other accounts.

In such a case, it would be convenient to create ten copies of the Container (one for each of the other ten accounts). If GoldSim allowed you to just make copies of the bank account Container, however, there would be confusion when you referred to an element by name (e.g., in an expression), since GoldSim would have no way of knowing which copy of the element you were referring to.

GoldSim’s solution to this problem is to allow you to localize portions of your model, such that any element names in the local region are hidden from elements outside of that region. A local region is referred to as a scope in GoldSim. Elements with the same name cannot exist within the same scope. Elements can, however, have the same name if they are located in different scopes. You change the scope of an element by localizing the Container in which it is located. Unless you specifically “expose” an element, the contents of a local Container can only be "seen" and hence referenced by other elements inside the Container.

The example model LocalizedContainer.gsm in the General Examples/Containers folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) contains an example of the use of Localized Containers.

Localizing a Container

All Containers in a model are either global or local. By default, new Containers that you insert into a model are global. You can localize a Container in two ways:

- By selecting Localize from the context menu for the element in the graphics pane or a browser (accessed via a right-click); or

- By selecting the Localization checkbox on the properties dialog for a Container.

You can recognize a localized container in four ways:

- The triangle in the upper left hand corner of the Container's symbol in the graphics pane is red (instead of blue).
The icon for the Container in the browsers is a closed box (rather than an open box).

The default symbol for the Container in the graphics pane is a closed box (rather than an open box):

- The tool-tip for a localized container will display “Localized”.

In addition, the properties dialog for the Container will indicate that it is localized.

If a Container is localized, then internally it is like a new model: all of the elements within it must have unique names. The elements within the localized Container, however, can share a name with elements outside of the Container.

Note: Like global Containers, localized Containers can be nested, so that a localized Container (referred to here as the parent) can itself contain a localized Container (referred to here as the child). In such a case, the scope of the parent localized Container does not extend into the child localized Container. Hence, an element ID within the child could be the same as one in some other location within the parent.

Read more: Nesting Localized Containers (page 948).

In addition to manually localizing a Container, GoldSim will also automatically localize Containers under some circumstances. In particular, whenever you paste a Container from one portion of your model (or one model) to another, GoldSim will automatically localize it if the names of elements inside the pasted Container conflict with those at the location where it is being pasted. (When it does this, it will warn you with a message.)

By design, the contents of a localized Container can only be "seen" (and hence referenced) by other elements inside the Container. Therefore, if you place an element called A in a localized Container called C, and then try to reference A in an expression (outside of C), GoldSim will report that it cannot find A (since it is "hidden" within the scope of the localized Container).

Eventually, however, you will need to pull information out of the localized Container to another portion of your model. GoldSim allows you to propagate outputs outside of localized Containers by exposing them as if they were outputs of the localized Container itself. Once an output is exposed, it is referenced in expressions by using the form ContainerID.OutputID.

For example, suppose you had a localized Container L, containing an Expression element named X. Assume further that another element named X existed outside of Container L. If you exposed X on the localized Container L, then you could access it from other Containers (outside of L) by referencing it as L.X in expressions.

There would be no confusion with the other element named X in the model, since you would be referring to the X within Container L as L.X (not simply X). Within Container L, however, you could access X in the normal way, by entering X directly in expressions.
An output can be exposed on a localized Container in four ways:

1. Whenever you localize a Container, any outputs which are already referenced outside of the Container are automatically exposed.

2. If you use the Link Cursor to create a link from an output within a localized Container to a location outside of the Container, the output is automatically exposed.

3. If you use the Insert Link dialog (accessed by selecting Insert Link… from the context menu of an input field) to create a link from an output within a localized Container to a location outside of the Container, you will be prompted with a message asking if you want to expose the output:

   If you select Yes, the output is exposed.

4. You can manually expose an output on a Container using the context menu for the output (accessed by left-clicking on the output port and then right-clicking on the output in the output interface).

   The dialog lists all localized Containers containing the element (i.e., if the element is contained within nested localized Containers, all of the localized Containers are shown). The output can be exposed on any (or all) of the localized Containers in which it resides by clicking on the Container name. If the output is already exposed on a particular Container, a check will appear next to the Container name.

Once an output is exposed, it remains exposed even if the element to which it was originally linked is deleted. The only way to unexpose an output is to access the context menu for the output, and under the Expose Output option, clear the check mark (by clicking on it) for the Container.
Note: Because moving an element is implemented as a cut and paste operation, and this operation effectively deletes and then recreates the links, if the move involves localized Containers, situations can arise where GoldSim will not be able to recreate all of the links. This can occur, for example, if you move an element into a localized Container such that elements referencing its outputs can no longer see them, or if you move an element outside a localized Container that references an output inside that Container. In such cases, you will have to expose the outputs (and press F9) in order to recreate the links.

Read more: Moving Elements Between Containers (page 102).

Once an output is exposed on a Container, it is displayed within an Exposed Outputs folder within the Insert Link dialog (accessed by right-clicking in an input field and selecting Insert Link…). For example, suppose that the variable X was exposed on the Container W. If you selected Insert Link… from an input field outside of the Container, you could select X from the Exposed Outputs folder:

Note that if the exposed output has a defined alias, this is shown first, followed by the element name in parentheses:
Localizing Containers

Read more: Defining an Alias for an Exposed Output (page 949).

Similarly, when typing output names into an input field, GoldSim’s link suggestion feature makes allows you to easily enter exposed outputs. In particular, if you wish to link to an exposed output of a localized Container, you can do so by typing a period after the name of the Container in the input field. The suggestion box will then list any exposed outputs that exist for that Container.

Read more: Displaying Link Suggestions in Input Fields (page 83).

Localized Containers can be nested, so that a localized Container (referred to here as the parent) can itself contain a localized Container (referred to here as the child). In such a case, the scope of the parent localized Container does not extend into the child localized Container.

To illustrate this, consider the following example. Suppose you had a localized Container L1, containing a second localized Container L2. A Reservoir element named R1, with an output (among others) called Overflow_Rate, exists in L2. Because this Reservoir is within a localized Container, a second Reservoir element with the same name could exist in L1.

If the Overflow_Rate for the Reservoir in L2 was exposed on L2, and the Overflow_Rate for the Reservoir in L1 was exposed on L1, these outputs would be referenced as shown schematically below:
As indicated in the diagram,

- The Overflow_Rate for the Reservoir within L2 would be referenced as:
  
  o R1.Overflow_Rate from within L2; and
  
  o L2.Overflow_Rate from within L1 (but outside of L2).

- The Overflow_Rate for the Reservoir within L1 would be referenced as:
  
  o R1.Overflow_Rate from within L1 (but outside of L2); and
  
  o L1.Overflow_Rate outside of L1.

Hence, if you were to reference R1.Overflow rate from an expression within L1, it would link to a different element than if you were to do the same from an expression within L2.

**Note:** In the example shown above, it is not possible to reference the Overflow_Rate for the Reservoir within L1 from within L2. In order to do so, you would need to first rename one of the two Reservoir elements.

---

**Defining an Alias for an Exposed Output**

There are some situations where referencing an exposed output of a localized container might be ambiguous. In particular, suppose you exposed the Overflow_Rate from the Reservoir in L2 on Container L1. How would you then reference this output from outside of L1? L1.Overflow_Rate is already used (it refers to the output from the Reservoir in L1). The same problem would occur if you added another Reservoir element with a different name (e.g., R2) to L1 and exposed the Overflow_Rate on L1. Again, L1.Overflow_Rate is already used and is not available.

**Read more:** [Search Logic for Linking to an Output Present in Multiple Scopes](page 951).
GoldSim solves this potential conflict by defining an alias for each exposed output. Whenever at least one output is exposed on a Container, an Exposed Outputs tab is added to the Container's property dialog:

If you hold the cursor over the Output field, a tool-tip showing the full path to the output is displayed.

The first column on this tab lists the full path of the output (which uniquely identifies it). The second column contains an alias. By default, the alias is the output name. When you reference an exposed output outside of a localized Container, it is not actually referenced as ContainerID.OutputID. Rather, it is referenced as ContainerID.Alias.

If you expose an additional output which has the same output name, GoldSim automatically appends a number to the alias.

In the example above, if you exposed the Overflow_Rate from Reservoir R2 on Container L1 after exposing the Overflow_Rate from Reservoir R1 on L1, then outside of L1, R2.Overflow_Rate would be referenced as L1.Overflow_Rate_2:

L1 (localized)

R1

refers to

L1.Overflow_Rate

R2

refers to

L1.Overflow_Rate_2
Note: The aliases automatically created by GoldSim for outputs with the same name depend on the order in which the outputs are exposed. The alias of the first output with a particular name that is exposed is always the output name. GoldSim then adds numbers (beginning with 2) to each additional output with the same name that is exposed on the Container.

Although GoldSim automatically creates aliases, you can manually edit an alias if you wish (by clicking on it in the dialog). The same rules that apply for element names apply for aliases: Aliases can only include letters, numbers, and the underscore (“_”) character. They cannot include spaces and must begin with a letter. Furthermore, for a given Container, no two exposed output aliases can be the same.

Although the outputs of an element in a localized Container can be referenced outside of the Container only if the output is exposed on that Container, the inputs to the element can come from any location in the model. That is, a localized Container is analogous to a two-way mirrored glass: you cannot look into the Container, but you can look out of it.

In cases where multiple outputs with the same name exist in a model, however, it is important to clearly identify the search logic used by GoldSim to link to outputs.

Consider the following example. Localized container L2 is contained within localized Container L1. Both L1 and L2 contain an element (with a primary output) named A1. A third element named A1 also exists outside of L1. Within L2, you create an Expression element (named B1), and specify its definition as 2*A1. Which of the three A1 elements does GoldSim create a link to?

The schematic above illustrates the search logic GoldSim would use to create the link. It would first look within its own scope (the local scope of B1). If it finds A1 there, it creates the link. If it does not find A1 in its local scope, it moves one level up in the scope hierarchy (in this case within L1) and looks for it there. If it finds A1 there, it creates the link. If it does not find A1 at that level, it continues to search upward through the various scopes that it has access to until it finds an A1, or exhausts the search (and fails to complete the link).
**Cloning Elements**

In some situations, you may wish to have elements in your model that always have identical definitions. You could accomplish this by copying an element to various locations in your model, but if you ever needed to change the definition of the element, you would need to make the change to all the copies, and this could be time-consuming and error-prone.

GoldSim addresses this issue by allowing you to create multiple clones of an element. When you create clones of an element, all of the clones behave identically: if you change one of the inputs to a clone, the same input is automatically changed for all of the other clones.

Clones are often used in conjunction with localized Containers. For example, suppose that you wished to model the salmon populations in each of ten streams. To do this, you might create ten parallel (adjacent) localized Containers, each of which represented a different stream. Although the inputs differ, let's assume that the basic algorithm used to compute the salmon population as a function of time was the same for each of the ten streams.

One way to build this model would be to create the elements representing the algorithm and copy them to each of the ten Containers. If, however, you were expecting to frequently change and modify the algorithm (e.g., as more data became available), it would become cumbersome (and error-prone) to edit the elements in all ten Containers whenever you wanted to make a change.

Clones provide an easier way to build and maintain such a model: You would simply create ten sets of clones (rather than copies) of the elements defining the algorithm and place them in each of the ten localized Containers. Then if you wanted to modify the algorithm, you would need only to change the elements in
Cloning Elements

one of the Containers and the corresponding clones of those elements would automatically be changed in all of the others.

Note that use of clones does not imply that all clones produce the same output result; it only implies that they have the same definition. For example, a cloned Expression element might have the definition 2*A. All clones of the element would indeed have the same definition, but if the clones were each located in a different localized Container, then A in one Container could have a different value than A in another Container (i.e., each localized Container could have its own element A). As a result, the output of the Expression would be different in each Container.

You create clones of an element by right-clicking on the element you wish to clone in the graphics pane or a browser to access its context menu, and selecting Clone Element.

When you do so, you will then be presented with a dialog containing all of the Containers in your model. Using this dialog, you must then select the Container into which you wish to place the clone.

If you select a Container in the same scope as the element you are cloning (or an existing clone of the element), GoldSim will ensure that the names of none of the clones conflict by adding a number to the end of the new clone. If you select a Container in a different scope, the cloned element will have the same name as the element being cloned.

All clones are "equal". That is, the original element is no different than the clone which was created from it. Both are clones, and if you change an input to one, the same input in the other is automatically changed accordingly.

You can always recognize a clone in the graphics pane, because a small symbol (two overlapping boxes) will be present in the upper right-hand corner of the image:

In addition, within the the properties dialog of a cloned element, you will notice an additional tab, labeled Clones.
Cloning Elements

The Clones tab lists all of the clones of the element. There is no limit on the number of clones of an element that can be created.

Double-clicking on a row in this dialog jumps directly to that clone.

When you move a cloned element (by right-clicking and selecting Move To…), the element remains a clone after it is moved.

However, if you copy and paste a cloned element, the pasted element is not a clone.

If you change an input for one clone, it is automatically changed for the other clones. This is the case for standard inputs, as well as other properties, such as checkboxes.

There are, however, several exceptions to this rule:

- Names (element IDs) are never cloned. Each clone can have a unique name.
- Descriptions are never cloned. Each clone can have a unique description.
- Save flags (i.e., Save Final Values and Save Time Histories) are never cloned. You must separately select whether you wish to save results for each clone.
- The graphical properties of an element (e.g., size, color, and symbol used in graphics pane) are not cloned. Each clone can have a different appearance.
- The random seed of Stochastic elements is not cloned. Therefore, cloned Stochastic elements have different seeds (and hence will have different sampled values).
- A number of elements (e.g., Reservoirs) provide an input field that only accepts Discrete Changes. These fields are followed by a button that allows you to more easily add multiple Discrete Changes. If an element is cloned, this button is disabled (such that the only way to add...
Cloning Elements

Multiple Discrete Changes is by separating them in the input field using a semi-colon).

**Elements that Cannot be Cloned**

Six element types cannot be cloned:

- Result elements (including Decision Analysis elements);
- Script elements;
- SubModels;
- CellNet Generators;
- File Elements; and
- Dashboards.

In addition, if a Data element is defined to provide Scenario Data, it cannot be cloned.

**Read more:** [Scenario Data: Defining Inputs for Different Scenarios](page 507).

No option is provided to clone these elements. Moreover, if they are present in a Container that you want to clone, you will not be able to clone the Container. You also cannot insert any of these elements into a previously cloned Container.

**Read more:** [Cloning Containers](page 955).

**Freeing a Clone (Decloning)**

In some situations, you may want to "declone" or "free a clone" (i.e., change it to a "normal" element). You can do this by right-clicking on the clone you wish to declone in the graphics pane or a browser to access its context menu, and selecting **Free Clone**.

In either case, the element will be changed into a "normal" element: it will no longer be a clone (and the Clone tab will no longer appear in the properties dialog for the element).

Note that if the element was one of only two clones before it was freed, the other remaining clone would also be automatically freed (since it would have no other clones).

Similarly, if you delete a clone which was one of only two clones before it was deleted, the remaining clone is automatically freed.

**Cloning Containers**

In addition to cloning individual elements, you can also clone an entire Container. Before cloning a Container, it must first be localized. Once a Container is cloned, it cannot be globalized.

When you clone a Container, all of its contents are also automatically cloned. Any change you make to any element within one cloned Container is made to all of the clones. In particular,

- When you create, paste or delete an element in one cloned Container, it is automatically created, pasted or deleted in all of the cloned Containers.
- When you move an element between Containers inside of a cloned Container, it is automatically moved within all of the cloned Containers.
- When you move an element into a cloned Container, the element is replicated and moved into all of the cloned Containers.
- When you move an element out of a cloned Container, that copy is moved and all of the clones of the element in the other cloned Containers are deleted.
Several other points regarding cloned Containers should also be noted:

- The names of all elements within a cloned Container are also cloned. (Recall that cloned elements which are not inside a cloned Container can have different names).
- You cannot declone an element which is inside a cloned Container.
- When you declone a Container, the elements inside the formerly cloned Containers remain clones. Once the Container is decloned, however, you can individually declone elements within the Container.
- Some elements cannot be cloned (e.g., Dashboards, Scripts, SubModels, Files, Results). As a result, GoldSim prevents your from adding these to a cloned Container, and prevents a Container that contains such elements from being cloned.
- If graphic images are present inside a Container when it is cloned, these are copied to the cloned Containers. If they are subsequently edited, however, the changes are not propagated to the other cloned Containers.
- When you change the positions of elements within a cloned Container (e.g., by dragging an element from one part of the screen to another), the new position is not propagated to the other cloned Containers.

The key to using cloned Containers is to provide unique inputs to them from elements that are *outside of them* (otherwise, all of the cloned Containers would behave identically and produce exactly the same results). This can be done by placing the cloned Containers within separate localized Containers:

In the example shown above, the two cloned Containers (Clone1 and Clone2) are within two separate localized Containers (L1 and L2). Within the cloned Containers, all the equations and elements are, by definition, identical. In this example, it is assumed that within each cloned Container, an element references Data1. Since the clones are in different localized Containers, however, they each link to a *different* Data1 element.

The example model CloningContainers.gsm in the General Examples/Containers folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) contains an example of the use of cloned Containers.

In general, you cannot copy a cloned element to create another cloned element. If you simply copy a clone and paste it elsewhere, the pasted element will not be a clone. The only way to clone an element is to right-click on it and select *Clone Element*.

Under one circumstance, however, you can create a clone by copying and pasting an element. In particular, you can create clones if you copy and paste a Container that contains clones.

When you copy and paste a Container that contains clones, GoldSim uses the following rules when the Container is pasted:
If the Container contains at least two elements that are clones of each other (cloned sisters), the pasted Container will also treat those elements as clones. Note, however, that they will not be clones of the elements in the original Container. For example, if Container1 contains elements A and B that are sister clones, and Container1 is copied and pasted (and the pasted Container is named Container2), A and B inside Container2 will be clones of each other, but they will not be clones of the original A and B in Container 1. The rule here is simple: when you copy and paste a Container with clones, under no circumstances can you increase the size of a set of existing clones. You can only make a copy of a set (forming a new set of clones).

If the Container contains only one clone (with the other clones being outside of the copied Container), the element will not be a clone in the pasted Container.

Referencing an Output's Previous Value

There are some situations in which you may want to reference the previous (as opposed to the current) value of an element’s output. For example, you may want to reference the amount of money in an account (represented by a Reservoir) one day ago, as opposed to at the current time. GoldSim allows you to reference an output’s previous value by using a Previous Value element.

A previous value element outputs the value of its input from the previous update of the model. Usually, a model is only updated every timestep, so that the previous update refers to the previous “scheduled” timestep specified in the Time Phase settings of the Simulation Settings dialog.

Read more: Setting the Basic Time Options (page 453).

In some cases, however, GoldSim actually updates the model between specified (scheduled) timesteps. In such a case, the previous value is not necessarily the previous scheduled timestep, and may return a value between the current scheduled timestep and the previous scheduled timestep.

Read more: Understanding Timestepping in GoldSim (page 455).

There are basically three instances in which a Previous Value element can be useful:

• Occasionally, you may actually want to directly reference the previous value of an output (as in the example of an account mentioned above). Generally, however, you should be careful when doing so, and in most cases it is poor modeling practice, as it ties your model to the timestep length. For example, if you were interested in obtaining the value for an output from 3 days previous, and to do so you used a Previous Value element, you would need to take care to always use a 3 day timestep. In such a case, rather than choosing a 3 day timestep and referencing its previous value using Previous Value element, it is almost always better to use an Information Delay with a Delay Time of 3 days.

Read more: Information Delay Elements (page 321).

• In some situations, you may wish to simulate a static or dynamic process in which the variables in the loop are coupled such that they respond instantaneously to each other, and there are no time lags. In GoldSim, these are referred to as recursive loops, and they are conceptually different from feedback loops. The output of a Previous
Value element can be used to provide an approximation to the current value of an output so that coupled equations can be solved explicitly.

**Read more:** Creating Recursive Loops Using Previous Value Elements (page 960).

- GoldSim provides the ability to carry out looping (iterative) calculations at each update (at each point in time) within a Container. In these special Containers, a calculation is repeated until a specified number of loops is completed or a particular condition is met. In most cases, the condition will likely involve comparing the output of the looping calculation to the previous loop’s output (e.g., determining if \((X_{\text{now}} - X_{\text{prev}}) < \text{some tolerance value}\)). Previous Value elements can be used to facilitate such a calculation.

**Read more:** Using Looping Containers (page 963).

### Inputs and Outputs to a Previous Value Element

The property dialog for a Previous Value element looks like this:

When defining a Previous Value element, your first step should be to specify the type of output for which you want to compute a previous value. You do this by pressing the **Type...** button, which will display the following dialog:

By default, the output of a new Previous Value element is a scalar, dimensionless value. However, a Previous Value element can accept as an input (and hence output) one of four types of outputs (as specified in the **Type** field in this dialog): Values, Conditions, Discrete Change Signals and Discrete Event Signals.

If you select Value, Condition, or Discrete Change Signal, the **Order** can also be specified as a vector or a matrix.
Referencing an Output’s Previous Value

The **Display Units** determine the dimensions of the input to and output from the Previous Value element, and are only applicable if the output is a Value or a Discrete Change Signal (otherwise the field is grayed out).

The **Reference** is the value for which you want to compute a previous value.

---

**Note:** If the **Reference** is a Discrete Event Signal or a Discrete Change Signal, it can accept multiple discrete signals. This is indicated in the input field by separating the individual discrete signals by semi-colons (e.g., Change1; Change2; Change3).

---

If the **Reference** is a Value or a Condition, in order to compute the output of a Previous Value element, GoldSim must know what value to use at the beginning of the simulation. This is specified within the **Initial Value** field.

---

**Note:** The **Initial Value** must be a number or a link from a static variable (e.g., a constant Data element, a Stochastic, or an Expression that is a function of constant elements).

---

Obviously, the **Reference** and the **Initial Value** inputs to the Previous Value element must have the same attributes (type, order and dimensions) as specified in the attributes dialogs (accessed via the **Type...** button).

---

**Note:** If you specify the output type as a Discrete Event Signal, the Reference field will also accept Discrete Change Signals. However, the information in the signal (Instruction and Value) will be lost when it is output by the Previous Value element.

---

If you place a Previous Value element inside a looping Container (and the Reference is a Value of a Condition), the element has an additional property (**Reinitialize when looping starts**) that controls how the Initial Value is treated between loops:

---

**Read more:** [Using Looping Containers](#) (page 963).

A Previous Value element has two outputs:

- The previous value of the Reference; and
- The rate of change of the Reference (i.e., the derivative).

The previous value is the primary output.

---

**Note:** The rate of change output is only available if the Reference is a Value.
Referencing an Output's Previous Value

The browser view of a Previous Value element is shown below:

![Browser View of Previous Value Element]

As can be seen, the browser view shows the two inputs (Reference and Initial Value) and the two outputs (the previous value itself, which has the same name as the element, and the RateOfChange).

**Note:** Element inputs and outputs are only shown in the browser if you choose to Show element subitems (accessed via the browser context menu by right-clicking in the browser).

Creating Recursive Loops Using Previous Value Elements

In some situations, you may need to define circular logic in your model that is not necessarily dynamic. A simple simultaneous system of equations is an example of such a system. Such logic does not represent a feedback loop, in that there is no chain of “cause and effect” and no dynamics – in fact, the variables may not change with time at all. In GoldSim, these kinds of circular systems are referred to as recursive loops (and are conceptually different from feedback loops).

**Read more:** Evaluating Feedback Loops (page 346).

As pointed out above, a simultaneous system of equations is an example of such a system. Consider the following example:

\[
\begin{align*}
B &= 4 - A \\
A &= (6 - B)/3
\end{align*}
\]

This is a circular system, because \( A \) is a function of \( B \) and \( B \) is a function of \( A \). To compute \( A \) and \( B \), it is necessary to solve this coupled system of equations. In this particular case, through simple substitution, you can convince yourself that the solution to these equations is \( A = 1 \) and \( B = 3 \).

If you encounter a system such as this in one of your models, you can handle it in one of two ways. First, you could solve the system of equations directly either prior to running the model (e.g., using substitution), or dynamically while running the model (e.g., using an External element or GoldSim’s matrix functions to solve the appropriate equations).

For simple systems such as the example above, this is obviously the correct approach. Note, however, that for many complex models (e.g., non-linear coupled equations), the solution could be very computationally intensive, requiring an iterative approach.

GoldSim therefore will allow you to specify such a circular system and solve the equations in an approximate manner. It does this by assuming, at every timestep, that the current value of one of the variables can be approximated by the value of that variable at the previous timestep. To solve the simple system outlined above, we would define the two equations in GoldSim as follows:

\[
\begin{align*}
B &= 4 - A \\
A &= (6 - \text{Previous}_B)/3
\end{align*}
\]
In the second equation, Previous_B is the output of a Previous Value element whose input is B:

By doing so, the equations can be solved explicitly (A is solved first as a function of the previous value of B, and then B is solved as a function of A). In effect, GoldSim uses the timestepping algorithm to iterate to a solution, as illustrated in the table below:

<table>
<thead>
<tr>
<th>Timestep</th>
<th>B at previous timestep</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>1.333</td>
<td>2.667</td>
</tr>
<tr>
<td>1</td>
<td>2.667</td>
<td>1.111</td>
<td>2.889</td>
</tr>
<tr>
<td>2</td>
<td>2.889</td>
<td>1.037</td>
<td>2.963</td>
</tr>
<tr>
<td>3</td>
<td>2.963</td>
<td>1.012</td>
<td>2.988</td>
</tr>
<tr>
<td>4</td>
<td>2.988</td>
<td>1.004</td>
<td>2.996</td>
</tr>
<tr>
<td>5</td>
<td>2.996</td>
<td>1.001</td>
<td>2.999</td>
</tr>
<tr>
<td>6</td>
<td>2.999</td>
<td>1.001</td>
<td>3.000</td>
</tr>
<tr>
<td>7</td>
<td>3.000</td>
<td>1.000</td>
<td>3.000</td>
</tr>
</tbody>
</table>

In this case, GoldSim iterated to a very good solution within 2 or 3 timesteps, and an almost exact solution in 7 timesteps. In order for this to work, however, it was necessary to specify the value of “B at the previous timestep” when time = 0. That is, it was necessary to define an Initial Value for B. For this example, an Initial Value of 2 was assumed.

This simple example model illustrating the use of Previous Value elements (PreviousValue.gsm) can be found in the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).
Note: A more effective way to solve this particular problem would be to embed the system of equations in a looping Container. In such a case, the iteration would not require any timestepping (the iteration would occur within a single timestep). In fact, looping Containers are one of the most important uses for Previous Value elements. This is illustrated in the file LoopingContainer.gsm, which can be found in the General Examples/Containers folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu). Alternatively, you could solve such a system of equations using a Script element. This is illustrated in the file Script.gsm, which can be found in the General Examples folder in your GoldSim directory.

Read more: Using Looping Containers (page 963); Script Elements (page 860).

When using Previous Value elements outside of looping Containers to represent recursive systems, you need to take special care to ensure that the length of the timestep you are using is small relative to the timescale over which the system is changing. If the system you are simulating is changing rapidly relative to the timestep, your computed result will be inaccurate. That is, if you are using the previous value of a variable as an approximation to the current value, and the previous value is in fact a bad approximation of the current value, then the computed result will be inaccurate (since it takes several timesteps to converge on the correct solution). Selecting a timestep for these kinds of simulations is discussed in further detail in Appendix F.

It is worth reiterating that recursive loops are conceptually different from feedback loops. Feedback loops represent a closed chain of cause and effect. Note that the terms “feedback” and “cause and effect” intentionally imply that the relationship between the variables is dynamic and the system changes over time (although systems with feedback loops can also reach a dynamic equilibrium). GoldSim allows you to create looping systems like this without having to use Previous Value elements. In order to do so, however, the loop must meet one requirement: it must contain at least one state variable.

Read more: Understanding State Variables in GoldSim (page 341).

Recursive loops, on the other hand, do not represent a chain of cause and effect, and the system may not be dynamic at all. These loops do not contain state variables. Simulating systems like these in GoldSim requires approximating the solution by using Previous Value elements (which actually work by inserting an “artificial” state variable into the loop).

When using the Function Of or Affects View on an element that is within a feedback or recursive loop, GoldSim will stop building a branch of the dependency tree as soon as an item is repeated (and it will mark this as a “loop”).

Read more: Viewing Element Dependencies (page 112).

For example, if A is a function of B, B is a function of the previous value of C, and C is a function of A, the Function Of View for A would look like this:
When using the Find function (Ctrl+F) in a Function Of or Affects view, one of the options is to search in Labels. If this is selected, and you enter “loop”, GoldSim will find the next loop in the list (since it will find the word “loop” in the label for the element). When using this search method, GoldSim will find all the loops (both feedback loops and recursive loops) in the model.

Read more: Finding Elements (page 112).

Using Looping Containers

In some models, you may want to carry out an iterative calculation at each timestep. This might be useful, for example, if you have a coupled system of equations that must be solved every timestep by iterating.

You can define a Container as a looping Container by selecting the Looping Capability feature in the Container dialog. When you do so, a Looping tab is added to the Container dialog:

Looping Containers are represented in the graphics pane as follows:
Using Looping Containers

**Note:** When you specify a Container as having **Looping Capability**, you cannot also define an **Internal Clock** for the Container (these two options are mutually exclusive).

**Note:** Looping Containers are useful when the looping calculation necessarily involves multiple elements (e.g., Reservoirs). For a calculation requiring simpler looping requirements (defining an array, or iterating to a solution for a simple equation), a Script element would often provide a more transparent and easier solution.

**Read more:** [Script Elements](page 860).

**Warning:** When you specify a Container as a looping Container, the **Treat as SubSystem** feature is also automatically selected (and cannot be deselected unless you first turn off **Looping Capability**). That is, a looping Container, by definition, is treated as a SubSystem. Because a looping Container is treated as a SubSystem, this puts certain limitations on how these Containers can be used.

**Read more:** [Treating a Container as a SubSystem](page 135).

Looping Containers are intended to be used to carry out an iterative calculation at each timestep. The number of loops carried out is controlled by specifying a condition (i.e., loop while or until a specified condition is met).

You can choose between two loop types:

- **Loop WHILE condition is true.** In this case, the loop is carried out as follows:
  1. At the beginning of the calculation, the loop counter is set to 1.
  2. GoldSim tests the **Condition**.
  3. If the Condition is False, the looping calculation is assumed to be complete.
  4. If the Condition is True, the contents of the Container are updated and the loop counter is incremented.
  5. Go to Step 2.

This is illustrated schematically below:

![Loop While True Logic Diagram](image)

**Controlling the Number of Loops in a Looping Container**
**Loop UNTIL condition is true.** In this case, the loop is carried out as follows:

1. At the beginning of the calculation, the loop counter is set to 1.
2. The contents of the Container are updated.
3. GoldSim tests the **Condition**.
4. If the Condition is True, the looping calculation is assumed to be complete.
5. If the Condition is False, the loop counter is incremented.
6. Go to Step 2.

This is illustrated schematically below:

![Loop Until True Logic Diagram]

**Note:** “Loop Until” Containers always update at least once. “Loop While” Containers may not update at all (if the loop condition is not met).

**Warning:** Because the Condition for a "Loop While" Container must be evaluated before the contents of the Container are evaluated, this Condition can only reference the loop counter or elements inside the Container that have well-defined initial conditions (i.e., state variables).

**Read more:** [Understanding State Variables in GoldSim](page 341)

The loop counter itself can be referenced by any element inside the Container (and within the **Condition** field) as “~LoopCount”. For example, if the **Condition** was defined as follows:

```
~LoopCount <= 10
```

then the Container would complete exactly 10 loops.
As indicated by the way it is referenced, the loop counter is an example of a locally available property. As such, it only has meaning inside the looping Container, and cannot be referenced outside of the Container.

**Read more:** [Understanding Locally Available Properties](page 808).

If you access the Insert Link dialog inside a looping Container (by right-clicking inside an input field of an element within the Container or in the **Condition** field itself), the loop counter appears in a **Available Properties** folder (displaying all of the locally available properties that can be accessed from that location):

The **Condition** must reference at least one output from within the looping Container (the loop counter is considered to be within the Container). If it does not, GoldSim will display an error message (since otherwise, the **Condition** could not change from loop to loop, and hence has no meaning).

---

**Warning:** GoldSim will not exit a loop until the **Condition** has been met, and there is no upper limit on the number of loops that can be carried out. As a result, you should be careful when defining your **Condition** to ensure that you do not define an infinite loop. You can always do so by directly referencing the loop counter (e.g., adding something to the **Condition** such as “OR ~LoopCount > 1000”) If it seems that you are in an infinite loop, you should pause the simulation (using the **Pause** button in the Run Control toolbar) and evaluate the loop counter and the condition. You can subsequently stop a simulation using the **Abort** button in the Run Control toolbar.

---

**Understanding How Elements Inside a Looping Container are Updated**

The elements inside the Container are updated as follows during the loops:

1. On the first loop, the elements are moved forward in time to the next scheduled timestep;
2. On subsequent loops, the elements are updated again without advancing time.

Hence, in order for the loops after the first to have any impact on the model, they must contain at least one state variable that changes with each loop.

**Read more:** [Understanding State Variables in GoldSim](page 341).

As an example, if a looping Container contained the simple system below, consisting of one Data element and one Expression (neither of which output a state variable), after the first loop, the system would not change at all:
Hence, it would serve no purpose to place a calculation such as this within a looping Container.

The most common application of a looping Container is to use it to carry out an iteration utilizing a Previous Value element. That is, some variable (e.g., $X$) would be computed each loop based on its value during the previous loop:

The looping system shown here would be updated every loop (changing $X$, and hence the previous value of $X$). Typically, you would specify that the looping continue until the difference between $X$ and the previous value of $X$ was within some specified tolerance.

When a Previous Value element is included inside a looping Container, it takes on a special property. In particular, an additional input option is added to the dialog that specifies how the Initial Value is to be treated:

If **Reinitialize when looping starts** is cleared (the default), the Initial Value specified in the field is only used at the beginning of the realization (when the Container is updated for the first time), and never again.

If **Reinitialize when looping starts** is checked, the value specified in the Initial Value field is used for the first loop every time the looping Container begins its calculations (i.e., it does not “remember” the value from the previous timestep).

*Read more:* [Inputs and Outputs to a Previous Value Element](page 958).

Another common application of a looping Container is to use it to carry out multiple discrete transactions in a single timestep. In the simple example below, items are moved from one Reservoir to another (via discrete changes) until the first Reservoir is empty:

*Read more:* [Modeling Discrete Changes to a Reservoir](page 386).

Examples of both of these types of applications (LoopingContainer.gsm) can be found in the Containers subfolder of the General Examples folder in your GoldSim directory (accessed by selecting **File | Open Example**... from the main menu).
Viewing and Modifying the Causality Sequence

When you build a model, GoldSim automatically sequences the elements in the order that they must be computed. For example if A was a function of B, and B was a function of C, C would be sequenced first, followed by B, followed by A. This is referred to as the causality sequence. Although in this simple example, the sequence is obvious, for complex models with looping logic, the causality sequence may not be readily apparent.

Read more: The Causality Sequence and Element Updating (page 343).

In most cases, you need not be concerned with how GoldSim sequences the elements. However, in some cases (e.g., when simulating systems that include discrete events and looping logic or systems that have competing Resource requirements), expert users may need to understand (and subsequently manipulate) the causality sequence.

You can view the causality sequence selecting Model|Causality Sequence from the main menu (or pressing F10). A dialog similar to this will be displayed:

The dialog indicates the order within the sequence (Pos), the element type (T), the Element ID and the full element path (Path).

There are actually two different sequences that can be displayed: the Static Sequence and the Function Sequence. The Static Sequence consists of those elements that only need to be computed once (at the beginning of the simulation), since they cannot change with time. The Function Sequence consists of those elements that can change as a function of time.

Read more: The Causality Sequence and Element Updating (page 343).

By default, the dialog displays the Function Sequence first (as this is typically of greatest interest). You can switch between these two using the drop-list in the top right corner of the dialog.
You can sort the rows according to any column (ascending or descending) by double-clicking on the column header.

The **Selection** field at the top of the dialog allows you to limit what portion of the sequence is displayed:

- All elements in the subsystem (the default)
- Elements in current container/child containers
- Elements in the current container only

**Note:** You can only view the causality sequence for one subsystem at a time. The subsystem that is displayed is the one you were viewing when you displayed the sequence.

**Read more:** [Treating a Container as a SubSystem](page 135).

Right-clicking on an element in the sequence displays the following menu:

Four of these options are self-explanatory (“Edit Element”, “Function Of”, “Affects”, and “Jump to Element”).

“Show Predecessors/Successors” provides a filtered view of the sequence showing only those elements that directly or indirectly either affect the selected element, or are affected by it. Note that when you choose this option, the position of the selected element is marked with a red background (making it easier to see):
“Specify Precedent” allows you to manually affect the causality sequence, and is discussed in detail in the next section.

Checking **Show causality sequence position in element tooltips** adds the position number to tooltips displays in the model:

---

**Note:** In some complex looping systems, you may notice that an Element ID in the sequence will be highlighted in red. This indicates that due to the looping nature of the system, GoldSim’s first choice for sequencing the element was not possible, and it was necessary to move the element upward in the sequence. GoldSim’s preference is to always have referencing elements follow the elements that they reference. When GoldSim is unable to generate a valid sequence using this rule, it selectively moves elements that reference state variables ahead of the elements that calculate the state variables, and flags the moved elements in red. The resulting sequence is not invalid, but it does indicate that the sequence is ambiguous. When you see a system like this, you should take special care to look carefully at the results to determine whether or not the system is performing as you expect it to. If not, you may need to manually change the sequence.
In order to determine how GoldSim is to carry out its calculations, it first must create the causality sequence for all of the elements. This represents the sequence in which the elements must be logically computed.

**Read more:** The Causality Sequence and Element Updating (page 343).

In some models, however, there may be more than one valid way to sequence the system, and these different sequences may produce different results. In these cases, GoldSim provides a way to manually force a particular causality sequence to ensure that the model accurately represents your system.

In order to better understand this, let’s construct a simple system where the causality sequence is ambiguous:

By default, the causality sequence for this model looks like this:

![Causality Sequence Diagram](image)

By default, the causality sequence for this model looks like this:

![Causality Sequence Table](image)

In order to fully understand this sequence, it is first necessary to recall a special rule regarding a Function Sequence involving Discrete Changes: Discrete Changes are propagated to any affected stock elements instantaneously when the Discrete Change is updated in the sequence (regardless of where the affected stocks are located in the sequence).

So in the sequence shown above, GoldSim does the following: Every timestep, after changing the system clock, GoldSim first computes the state variables that are intrinsic functions of time. In this example, the only element that falls into this category is the Reservoir, so it is first brought up to the current time. After doing this, GoldSim then computes the Function Sequence. First, it computes the value of the Discrete Change, and instantaneously applies the Discrete Change to the Reservoir (this is actually regardless of where the Reservoir is in the Sequence). Then it computes the Reservoir (i.e., it would store all of its inputs for use in the next timestep, but this would not change its current value). Finally, the Expression is computed.

The reason this system is ambiguous is related to the fact that the Reservoir is actually computed twice. First, it is brought up to the current time (i.e., it solves a time integral). Then, it is updated a second time within the Function Sequence (it is modified by the Discrete Change). The ambiguity here arises from the fact that it is not completely clear when you might want to compute the Expression.

There are two options:

1. Compute the Expression after the Reservoir has been updated by the Discrete Change; or
2. Compute the Expression after the Reservoir has been brought up to the current time, but before it has been updated by the Discrete Change. GoldSim will sequence it assuming the first option is correct. However, the second option is equally valid, and will produce a different result! Hence, when you have a system such as this, you must specifically instruct GoldSim how to sequence this system to ensure it accurately represents the system you are trying to model.

To facilitate this, GoldSim provides an option to manually force a particular causality sequence. If you right-click on an element in the sequence, one of the options is to “Specify Precedent”. You can then specify a particular element that you wish the selected element to follow in the causality sequence. In this particular case, to force the second option listed above, we want the Expression to be a precedent for the Discrete Change. Right-clicking on the Discrete Change and selecting “Specify Precedent” displays this dialog:

If we select Expression here, the causality sequence now looks like this:

Note that Expression is now in front of the Discrete Change. The Discrete Change is highlighted in gold (indicating that it has a “forced” precedence). The input port for the Discrete Change is also highlighted in gold, and its precedence requirement is added to its tool-tip:
If you right-click on the Discrete Change, you would see that there is a new option (“Remove Precedent”).

Two other points regarding forced precedence should be noted:

- You can only specify one precedence requirement for an element (although an element can have multiple elements that use it as a precedent).
- GoldSim will not allow you to specify a precedent that results in an invalid causality sequence (e.g., a recursive system). It only allows you to force sequences that are still valid.

An example of the use of forced precedence to control sequencing (Bounce.gsm) can be found in the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

**Using SubModels to Embed Models Within Models**

In some cases, you may want to embed an entire GoldSim model within another GoldSim model. GoldSim provides a special element called a SubModel to facilitate this.

A SubModel superficially looks like a Container, and conceptually shares some aspects with Containers. However, the functionality of a SubModel is quite different from a Container. Whereas a Container is simply a grouping of elements within a model, a SubModel is a completely separate “inner” model within an "outer" model. That is, it has its own simulation settings (time and Monte Carlo options) that are independent of the simulation settings of the outer GoldSim model within which the SubModel element is placed. Hence, when a SubModel element (i.e., the inner model) is triggered to do a calculation by the outer model, it runs a complete simulation.

The sections below provide the details on the use of SubModels in GoldSim.

Before describing the details of how SubModels are created and used, it is worthwhile to first discuss why you might want to use a SubModel.

Recall that a SubModel is a complete "inner" model that has been embedded in an "outer" model. The inner model and the outer model can take on a number of forms (e.g., static/dynamic, deterministic/Monte Carlo, simulation/optimization). This provides a wide variety of potential uses. The most common are summarized below.

**Manipulation of Monte Carlo Statistics.** In some cases, after carrying out a Monte Carlo simulation, you may want to carry out further calculations using the statistical outputs of the simulation. For example, you may want to carry out a calculation that is some function of the mean and the 95th percentile of a particular output in a Monte Carlo simulation. Without the use of SubModels, the only way to accomplish this is by exporting results
from a Monte Carlo simulation manually to another application (e.g., a spreadsheet or a separate GoldSim model). With SubModels, this is easily accomplished within a single GoldSim model by inserting a SubModel (that performs a Monte Carlo simulation) into an outer model (that is static and simply manipulates the statistical outputs of the inner model).

**Explicit Separation of Variability from Uncertainty.** Many models have uncertain parameters as well as randomly variable parameters. An uncertain parameter represents ignorance that can theoretically (but perhaps not practically) be reduced through investigation (e.g., the mean failure time for a batch of light bulbs). Variability is inherent in many systems (e.g., the distribution of failure times for a batch of light bulbs) and cannot be reduced. It is often valuable to explicitly separate variability from uncertainty in a model. With SubModels, this is accomplished by inserting a SubModel (e.g., a dynamic Monte Carlo simulation) within an outer model (e.g., a static Monte Carlo simulation). This is referred to as "nested" Monte Carlo simulation. In the example above, the outer model would sample a probability distribution that represents the uncertainty in the mean lifetime of a light bulb, and the inner model would simulate the performance of a number of random light bulbs (whose lifetime is sampled from a distribution with the mean specified by the outer model). The result of this type of analysis is a probability of probabilities.

**Probabilistic Optimization.** If you wish to optimize a probabilistic (uncertain) system, the objective function to be optimized cannot be a single deterministic output. Rather, it must be a statistic. That is, if X was an output of a probabilistic model (and hence was output as a probability distribution), optimizing X itself would be meaningless. Rather, you would need to optimize a particular statistic (e.g., the mean or 50th percentile) of the output X. With SubModels, this is accomplished by inserting a SubModel (e.g., a dynamic Monte Carlo simulation) within an outer model (e.g., a static optimization).

**Dynamic Optimization During a Simulation.** Imagine a situation where you were simulating the operation of a facility over a period of one year. Every month, the operators make a decision based on the current state of the system. This decision is based on a simple optimization analysis using currently available data (i.e., at every month during the simulation). The optimization chooses the optimum values of a few control variables that they will use for the next month. With SubModels, you could simulate this by inserting a SubModel (e.g., a static optimization) within an outer model (e.g., a dynamic simulation).

Example files describing all four of these applications (SubModel1.gsm, SubModel2.gsm, SubModel3.gsm, and SubModel4.gsm) can be found in the General Examples folder of your GoldSim directory (accessed by selecting **File | Open Example...** from the main menu) in the subfolder named SubModels.

**Read more:** SubModel Examples (page 1011).

**Creating a SubModel**

SubModels are listed under Container in the Insert Element context menu:
When you insert a SubModel, you are presented with a dialog asking if you want to create a new (empty) SubModel, or create the SubModel by importing an existing standalone model:

If you choose to import a standalone model, you will be prompted for the file name and a SubModel with the contents of that model will be inserted into GoldSim. Otherwise, a new (empty) SubModel will be created.

Read more: Importing SubModels (page 1007).

The default icon for the SubModel element looks like this:

The property dialog looks like this:

Like all GoldSim elements, you first specify an Element ID and a Description. After inserting the SubModel element, the steps for building a SubModel are generally as follows (although they will typically be done iteratively, rather than in this exact order):
1. Specify the Solution Type (Simulation or Optimization).
2. Specify (or if an existing model was imported, edit) the GoldSim modules required by the SubModel.
3. Build (or if an existing model was imported, edit) the contents of the SubModel.
4. Specify (or if an existing model was imported, edit) the simulation settings for the SubModel.
5. Create an input interface between the SubModel and the parent (outer) model.
6. Create an output interface between the SubModel and the parent (outer) model.
7. Specify when the SubModel is to be run.

These steps are described in detail in the sections below.

Specifying the Solution Type for a SubModel

The first step in creating a SubModel (after inserting the element) is to specify the Solution Type. There are two options: "Simulation" and "Optimization".

This option determines how the SubModel is being used. For most applications, you should choose "Simulation", which is the default.

"Optimization" will normally only be used when a SubModel is being used to carry out a dynamic optimization (i.e., at specified times) during a simulation. For example, imagine a situation where you were simulating the operation of a facility over a period of one year. Every month, the operators make a decision based on the current state of the system. This decision is based on a simple optimization analysis using currently available data (i.e., at every simulated month during the simulation). The optimization chooses the optimum values of a few control variables that they will use for the next month.

To represent this in GoldSim, you would need to specify the Solution Type for the SubModel as "Optimization", and the SubModel itself would represent the optimization calculations carried out by the simulated operator.

Note: If the Solution Type is “Optimization”, Monte Carlo options for the SubModel must be set to “Deterministic”.

Specifying the Modules and Module Options for a SubModel

Read more: Specifying the Simulation Settings for a SubModel (page 978).

If you do choose the Simulation Type as "Optimization", you will also need to specify the optimization settings in the Optimization tab for the SubModel.

Read more: Running an Optimization Within a SubModel (page 1003).

Because a SubModel is effectively a separate model (as opposed to a Container), you may choose to use different extension modules in a SubModel than in the outer model. For example, your SubModel may require the Contaminant Transport Module, while the outer model may not.

When you create a SubModel, the modules that are activated by default are the same as those that would be activated by default if you created a new model. (These defaults are set by the user by pressing the Set as Default button when activating modules in the Modules tab of the Options dialog).

Read more: Activating and Deactivating Extension Modules (page 18).
To specify the modules that will be activated for a SubModel, press the **Modules...** button on the SubModel dialog. The following dialog will be displayed:

![Options dialog](image)

All extension modules that you are licensed to use appear in the dialog. If your license does not allow you to use a particular module, it will not be listed. You can activate and deactivate modules that you are licensed to use by clicking the **Active** checkbox.

If you deactivate a module, the specialized elements associated with that module will be deleted from your SubModel (if any are present) and any associated menu options will be removed in the current file. If you make a module active, the various options associated with that module are made available again.

**Note:** If a module that is active in a SubModel has special options (that are normally accessed via a tab in the **Options** dialog for a standalone model), these options will appear in a tab on the Modules dialog for the SubModel.

---

### Building the Contents of the SubModel

Once you have selected the Solution Type and modules to be used by a SubModel, building the contents of a SubModel is, for the most part, identical to building the contents of a Container.

Like a Container, you enter a SubModel by clicking on the plus sign in the upper left hand corner of the element. SubModels (and their contents) are also displayed in the browser in the same manner as Containers.

You can add elements, Containers, and even other SubModels to a SubModel, just as you would to a Container. Of course, you can also copy elements that are outside of a SubModel and paste them into the SubModel.

**Note:** In addition to creating a model manually inside a SubModel, you can also choose to import an existing (standalone) model into a SubModel.

**Read more:** [Importing SubModels](page 1007).
There are several important points to note when building and navigating a SubModel:

- When you are inside a SubModel, the navigation bar at the top of the graphics pane only shows the location with respect to the SubModel. It does not show the path relative to the parent model:

  ![SubModel Navigation](image)

- Within a SubModel, you cannot reference outputs that exist outside of the SubModel in the same way you would from inside a Container. That is, by default, the SubModel is a self-contained system (i.e., a separate model) that cannot "see" anything on the outside. In fact, within a SubModel, the Insert Link dialog accessed via the context menu for input fields does not list any elements outside of the SubModel. In order to access outputs from outside the SubModel, you need to take some specific actions (i.e., add the output to the SubModel's input interface).

  Read more: [Creating the Input Interface to a SubModel](page 980).

- Because the SubModel is a self-contained system (i.e., a separate model), elements on the outside cannot directly reference outputs inside the SubModel. Of course, in order to be of any value, a SubModel must have some way to communicate with (i.e., provide outputs to) the outer model. However, in order to provide outputs from inside the SubModel to elements in the outer model, you need to take some specific actions (i.e., add the outputs to the SubModel's output interface), and the SubModel outputs themselves have added complexity (since they may represent complex results, such as a distribution resulting from a Monte Carlo simulation).

  Read more: [Creating the Output Interface to a SubModel](page 984).

- Results inside a SubModel behave differently than results that have been added to the SubModel’s output interface. In particular, because a SubModel is a separate simulation (that will typically be carried out multiple times during a simulation of the outer model), results inside of a SubModel are typically overwritten during a simulation of the outer model. As a result, although you can choose to save the results inside of a SubModel, only results from the last simulation of the SubModel (i.e., the last time the SubModel was run) are available for viewing inside the SubModel at the end of your simulation. Note, however, that this does not apply to results that have been added to the SubModel’s output interface. A wide variety of complex results can be accessed via the interface, including various statistics and raw data for all SubModel time histories for all simulations of the SubModel.

  Read more: [Saving and Viewing Results Inside a SubModel](page 991); [Creating the Output Interface to a SubModel](page 984).
The inputs to be entered here are identical to those provided in the **Time** tab of the Simulation Settings dialog for a model.

**Read more**: Setting the Basic Time Options (page 453).

The **Monte Carlo** tab is used to specify the Monte Carlo options for a simulation:
With one exception, the inputs to be entered here are identical to those provided in the Monte Carlo tab of the Simulation Settings dialog for a model.

**Read more:** Setting the Monte Carlo Options (page 479).

The one exception is the **Use a different random number seed for each realization of the parent model** field. If this option is checked (the default), the Monte Carlo SubModel will behave randomly, producing a different result for each realization of the parent model and for each time the SubModel is calculated. If this box is cleared, the Monte Carlo SubModel will behave identically for each realization of the parent model, so that, for example, realization 17 of the inner model is the same for every realization of the parent model.

**Note:** You can choose to specify some of the simulation settings for a SubModel via the Input Interface for the SubModel. If you do this, some of the input fields in the Time and Monte Carlo tabs will become locked.

**Creating the Input Interface to a SubModel**

Because a SubModel is a self-contained system (i.e., a separate model), elements inside a SubModel cannot "see" anything on the outside (i.e., in the outer model). As a result, you cannot reference outputs that exist outside of the SubModel in the same way you would from inside a Container. (In fact, within a SubModel, the Insert Link dialog accessed via the context menu for input fields does not list any elements outside of the SubModel).

Of course, in order to be of any value, a SubModel must have some way to communicate with (e.g., access outputs of) the outer model. This is done by creating an interface between the SubModel and the outer model.

The interface is accessed via the **Interface** tab on the SubModel dialog:
The top part of this dialog is used to define the input interface. The bottom portion is used to define the output interface.

In order to access an output outside of the SubModel, you must add the output to the input interface. This is done by pressing the Add button in the Input Interface portion of the dialog. When you do this, GoldSim displays the Insert Link dialog, allowing you to select an output outside of the SubModel to add to the interface.

After selecting an output, the following dialog is displayed:

The **Name (ID)** is the name by which the output variable can be referenced inside the SubModel. This has the same restrictions as an element ID. The **Description** is an optional description of the variable.

You must then specify the **Display Units** and **Type** information (e.g., value/condition, scalar/vector/matrix) for the variable. Finally, you specify the **Definition** for the input. Typically, this will be a link to an output in the outer model that you selected. However, it can also be an equation (with links), or a constant value.
In addition to Values and Conditions, three types of complex outputs can be specified as inputs to a SubModel Input interface: Time Series Definitions, Lookup Table definitions, and Distribution definitions. These are specified via the Type button. In the two former cases, additional information (e.g., what the Time Series represents, units for the independent variables for a Lookup Table) must also be specified in the input interface dialog.

A Time Series definition can be produced by a Time Series element, an External element or another SubModel. A Lookup Table definition can be produced by an External element. A Distribution definition is one of the outputs of a Stochastic element (and can also be produced by another SubModel). Passing these definitions through the input interface allows you to pass complete time series, lookup table and distribution definitions into the SubModel, where they can be used to define Time Series, Lookup Table or Stochastic elements.

Read more: Referencing a Time Series Definition Output (page 225); Using an External Element to Define Lookup Tables (page 937); Externally-Defined Distribution (page 167).

Once you have added an output from the outer model to the Input Interface of a SubModel, you can subsequently reference the output within the SubModel by using the specified Name that you defined previously, preceded by a ~. For example, if you create an interface input called X, you could reference it in input fields within the SubModel as ~X.

You can also use the Insert Link option to reference an item on the Input Interface by right-clicking in an input field within the SubModel. When you do so, all of the items on the Input Interface to the SubModel will be displayed within the "Available Properties" folder:

Note: In this case, the ~ indicates that this output has some special characteristics (in particular, it is coming from outside of the SubModel). This is an instance of what is referred to as a locally available property in GoldSim. Locally available properties are accessed by using a ~.

Read more: Understanding Locally Available Properties (page 808).

In some cases, you may want to control some of the simulation settings (i.e., time and Monte Carlo options) of the SubModel from outside of the SubModel (e.g., using values provided by the outer model). To facilitate this, GoldSim allows you to add run properties to the input interface.
This is done by pressing the Add Run Property button in the Input Interface portion of the dialog. When you do this, the following dialog is displayed:

![Add Run Property button]

The Run Property field provides a drop-list of options for the SubModel simulation setting that you wish to control. The options are:

- Simulation Duration;
- Maximum time between updates;
- Number of Realizations; and
- Realization to Run

The Definition field determines the value that will be used for the selected simulation setting input. Typically, this will be a link to an output in the outer model (the Definition field's context menu can be used to access an Insert Link dialog that lists all elements in the outer model). However, it can also be an equation (with links), or a constant value.

**Note:** If you wish to control the Number of Realizations Run Property, the value must always be set to a number greater than 1. That is, you cannot switch between a single realization run and a multi-realization run of the SubModel dynamically via the Input Interface.

Note that when you select one of these options, the appropriate input fields are automatically locked (and cannot be edited) in the **Time** and **Monte Carlo** tabs for the SubModel:
Using SubModels to Embed Models Within Models

Because a SubModel is a self-contained system (i.e., a separate model), by default, elements outside a SubModel cannot "see" anything on the inside. Moreover, because a SubModel is a separate simulation (that will often be carried out multiple times during a simulation of the outer model), intermediate results of a SubModel are typically overwritten during a simulation of the outer model and are not available in the outer model.

Of course, in order to be of any value, a SubModel must have some way to communicate with (e.g., provide outputs to) the outer model. This is done by creating an output interface between the SubModel and the outer model.

The output interface is accessed via the Interface tab on the SubModel dialog:

**Creating the Output Interface to a SubModel**

Note: Buttons for deleting, editing, moving and alphabetically sorting the inputs on the interface are available directly below the Add and Add Run Property buttons.
The top part of this dialog is used to define the input interface. The bottom portion is used to define the output interface.

The output interface allows you to specify the outputs that you want the SubModel to provide to the outer model. It is important to understand that the output interface represents the only mechanism by which you can access outputs of a SubModel outside of the SubModel.

In order to allow the outer model to access an output of the SubModel, you must select the output and add it to the output interface. This is done by pressing the Add button in the Output Interface portion of the dialog.

When you press this button, you will be presented with a browser displaying the outputs inside the SubModel. You must select the output that you are interested in.

After you select an output, the following dialog will be displayed:

The Output ID is the name by which the output variable can be referenced in the outer model. This ID has the same restrictions as an element ID.
Note that the outputs defined in the output interface are added to the output port interface of the SubModel and can be accessed there (in exactly the same manner the outputs from other kinds of elements are accessed). The Output ID is what is shown in the interface:

The **Description** is an optional description of the output. The **Internal Output** field displays the selected output (which you can change using the button to the right of the field).

The **Data Type** and **Unit** fields are automatically populated based on the **Internal Output** selection.

After selecting the **Internal Output** you must then select what type of result is to be made available to the parent model on the output interface. This is controlled by two fields: **Output Type** and **Output Definition**. For some **Output Definition** selections, a third field will be presented to the immediate right of the **Output Definition** field.

There are two options for **Output Type**:

- **Final Value**. The following **Output Definitions** are available (some of which are only valid selections under specific circumstances):
  - **Last Calculated**. This selection is always valid.
  - **Statistic**. This selection is only valid for Probabilistic SubModels (it is not valid for Deterministic SubModels). A specific Statistic must be defined immediately to the right of the **Output Definition** field.
  - **Distribution**. This selection is only valid for Probabilistic SubModels and scalar outputs (it is not valid for Deterministic SubModels or for array outputs).

- **Time History**. The following **Output Definitions** are available:
  - **Last Calculated**. This selection is only valid for dynamic SubModels and scalar outputs (it is not valid for Static SubModels or for array outputs).
  - **Statistic History**. This selection is only valid for Probabilistic, dynamic SubModels and scalar outputs (it is not valid for Deterministic or Static SubModels or for array outputs). A specific Statistic must be defined immediately to the right of the **Output Definition** field.
  - **Realization Histories**. This selection is only valid for Probabilistic, dynamic SubModels and scalar outputs (it is not valid for Deterministic or Static SubModels or for array outputs).
**Note:** For some advanced applications, you may select a Time Series Definition output (from an External element or Time Series element inside the SubModel) rather than a Value as the **Internal Output**. If you do so, the **Output Type** is automatically set to be a Time Series Definition (rather than a Final Value or Time History).

When you add an output to the Output Interface and select the **Output Type** and **Output Definition**, you will be notified if your selections are invalid.

The Output Interface provides a summary of all of the outputs (as well as their **Output Type** and **Output Definition**):

<table>
<thead>
<tr>
<th>Name</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow_Rate_Final</td>
<td>Final Value</td>
</tr>
<tr>
<td>Flow_Rate_Mean</td>
<td>Statistic (Mean)</td>
</tr>
<tr>
<td>Flow_Rate_Distribution</td>
<td>Final Value Distribution</td>
</tr>
<tr>
<td>Flow_Rate_History</td>
<td>All Realization Histories</td>
</tr>
<tr>
<td>Flow_Rate</td>
<td>Time History</td>
</tr>
</tbody>
</table>

If you change the settings such that some of the selections are no longer valid, the Results will be identified in red (and the model will not run). In the example below, the SubModel has been set to Deterministic, invalidating those Output Types that require a Probabilistic SubModel:

<table>
<thead>
<tr>
<th>Name</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow_Rate_Final</td>
<td>Final Value</td>
</tr>
<tr>
<td>Flow_Rate_Mean</td>
<td>Statistic (Mean)</td>
</tr>
<tr>
<td>Flow_Rate_Distribution</td>
<td>Final Value Distribution</td>
</tr>
<tr>
<td>Flow_Rate_History</td>
<td>All Realization Histories</td>
</tr>
<tr>
<td>Flow_Rate</td>
<td>Time History</td>
</tr>
</tbody>
</table>

It is important to understand what these various results are and how they can be used in the parent model:

**Final Value Results**

**Last Calculated.** This result is simply the final value of the output that was computed at the end of the simulation (for the final realization of the SubModel if multiple realizations were run). This will be listed in the Output Interface as “Final Value”. It can be used directly in the parent model as any other value (or condition) output could be used.

**Statistic.** If this is selected as the Output Definition, a third field appears directly to the right:
The first two options are “Mean” and “50%” (the Median). The third option provides access to a dialog where you can specify any percentile. In these three cases, the result represents the specified statistic (computed over all realizations of the SubModel) for the final value of the selected output (i.e., the value computed at the end of each realization).

The fourth option (“Specify Result Condition”) provides access to the following dialog:

[Image of a dialog box showing probability calculation]

In this case, the result represents the probability (computed over all realizations of the SubModel) that the final value of the output satisfies the specified condition. In the example above, within the parent model, the result would represent the probability that the final value of the selected SubModel output was less than 0 m³/day.

In all cases, the result will be listed in the Output Interface as “Final Value (Statistic)”, where Statistic is the selected option. It can be used directly in the parent model as any other value (or condition) output could be used.

**Distribution.** This result is a complex output (referred to as a Distribution output) that represent all the statistical information necessary to define a probability distribution computed by the SubModel. It can be used to define a Stochastic in the parent model (as an “externally-defined” distribution). More frequently, it will serve as an input to specialized functions that operate on distributions. For example, PDF_Mean(X) returns the mean of the distribution X; PDF_Value(X,0.95) returns the 95% percentile. Under certain conditions (nested Monte Carlo simulation), Final Value Distributions can represent a distribution of distributions.

The result will be listed in the Output Interface as “Final Value Distribution”. Unlike other results, because it is a complex type of output, it can only be used in the parent model in specific locations (e.g., as input to an externally-defined Stochastic, as input to a specialized distribution function, within a Distribution Result element).

**Read more:** [Externally-Defined Distribution](#) (page 167); [Specialized Functions That Operate on Distributions](#) (page 182); [Adding a Distribution Output to a Distribution Result](#) (page 663); [Carrying Out Nested Monte Carlo Simulation Using a SubModel](#) (page 992).

**Time History Results**

Time History results on a SubModel Output Interface are complex outputs that represent all the information necessary to define a time series. As such, they can only be used in three places within the parent model: 1) to define a Time Series element in the parent model; 2) as an input to an External (DLL) element; or 3) directly within a Time History Result.

**Read more:** [Referencing a Time Series Definition Output](#) (page 225); [Using an External Element to Read and/or Output Time Series](#) (page 939); [Viewing SubModel Results in Time History Result Elements](#) (page 617).

**Last Calculated.** This result is the time history of the output for the final realization of the SubModel (which is equivalent to the only realization if the
SubModel Monte Carlo Settings do not specify multiple realizations). Hence, regardless of whether the SubModel is probabilistic or deterministic, it always consists of a single time history (for each realization of the parent model). This will be listed in the Output Interface as “Time History”.

**Statistic History.** If this is selected as the Output Definition, a third field appears directly to the right:

The first two options are “Mean” and “50%” (the Median). The third option provides access to a dialog where you can specify any percentile. In these three cases, the result represents a time history of the specified statistic (computed over all realizations of the SubModel) for the selected output.

The fourth option (“Specify Result Condition”) provides access to the following dialog:

In this case, the result represents a time history of the probability (computed over all realizations of the SubModel) that the output satisfies the specified condition. In the example above, within the parent model, the result would represent the time history of the probability that the selected SubModel output was less than 0 m³/day.

In all cases, regardless of whether the SubModel is probabilistic or deterministic, it always consists of a single time history (for each realization of the parent model). This will be listed in the Output Interface as “Statistic History ([Statistic])”, where **Statistic** is the selected option.

**Realization Histories.** This result is the time history of the output for all realizations of the SubModel. Hence, unless the SubModel Monte Carlo Settings do not specify multiple realizations, it consists of multiple time histories (for each realization of the parent model). Hence, when running nested Monte Carlo simulation (i.e., in which both the parent model and the SubModel are probabilistic), this can result in a very complex set of results. For example, if the parent model was run for 100 realizations, and the SubModel was run for 50 realizations, the results being displayed in a Time History Result element within the parent model would be based on 5000 individual time histories (50 * 100). Fortunately, GoldSim provides some powerful tools to view this complex set of results.
Using SubModels to Embed Models Within Models

Read more: Carrying Out Nested Monte Carlo Simulation Using a SubModel (page 992); Viewing SubModel Results in Time History Result Elements (page 617).

This type of result will be listed in the Output Interface as “All Realization Histories”.

Note: Calculations in GoldSim are carried out in double-precision. However, outputs of a SubModel are reduced to single precision when they pass through the output interface.

Note: Buttons for deleting, editing, moving and alphabetically sorting the outputs on the interface are available directly below the Add button.

At the bottom of the Definition tab for SubModels are options to Save Results for Final Values and Time History. The outputs for a SubModel, of course, are those results that have been added to the output interface. However, it is important to understand that these flags do not apply to all outputs in the output interface. In particular, they only apply to “standard” outputs (which must be Last Calculated or Statistic Final Values). They do not apply to the special output types discussed above (Distribution outputs or Time Series Definition outputs). These output types can only be viewed through Result elements in the parent model. (Note that time histories of “standard” SubModel results as viewed in a dynamic parent model would only change if the SubModel was called multiple times during simulation of the parent model).

By default, a SubModel is treated like any other element in GoldSim and is updated (i.e., run) whenever required to do so by the parent model (e.g., when inputs change).

In some cases, however, you may want to manually control when a SubModel is updated. You can do so by checking the Update (run) SubModel only when triggered checkbox. When you do this, access is provided to a standard trigger dialog:

You can explicitly control when a SubModel is run by defining one or more triggers (e.g., “On True: X > Y”).

Read more: Understanding Event Triggering (page 355).
By default, when this box is checked, no triggers are defined. This means that the SubModel will never run.

**Note:** You can tell if a trigger has been defined from the appearance of the Trigger... button. If a trigger is defined, the rectangle next to the lightning bolt is bright green; otherwise it is dark green. And like all Trigger... buttons, it displays a tool-tip. If there is no trigger defined, the tool-tip will display “Never updates SubModel”.

If you wish the SubModel to run when its parent Container activates, set the update trigger to **Auto Trigger**. Note that if a SubModel is set to Auto Trigger and it is not within a conditional Container, it activates when the Model (root) Container activates (i.e., at the start of the realization).

Prior to a SubModel being triggered, its outputs are all zero. The outputs are updated when the SubModel is triggered, and outputs remain constant until the SubModel is triggered again.

Several advanced options are available for SubModels. These are discussed in the sections below.

Because a SubModel is a separate simulation (that will often be carried out multiple times during a simulation of the parent model), results inside of a SubModel are typically overwritten during a simulation of the parent model.

Accordingly, by default, other than results passed through the SubModel's output interface, no results are actually saved (and hence, you cannot view results inside a SubModel after running a simulation).

However, you can choose to save (and view) the results of the last simulation of the SubModel. This can be done by checking the **Save Results for most recent Simulation** checkbox on the SubModel's **Definition** tab:

If this box is checked, results from the last simulation of the SubModel (i.e., the last time the SubModel was run) are available for viewing inside the SubModel at the end of your simulation. Note that the SubModel's results could represent
Using SubModels to Embed Models Within Models

multiple realizations, and in such a case, both time history and probabilistic results would be available for viewing inside of the SubModel. However, if the SubModel was run multiple times by the outer model (e.g., every timestep of the outer model), only the results from the last time the SubModel was run would be available.

Note: There are two important exceptions to this. In particular, under certain circumstances, Result elements in the parent model can display multiple realizations from the SubModel: 1) If you are running a nested Monte Carlo simulation (both the SubModel and the parent model are running multiple realizations), Distribution Result elements in the parent model modify their behavior to enable you to view multiple sets of realizations produced by the SubModel. 2) You can select a special option in Time History Result elements in the parent model to enable you to view multiple time histories from within a SubModel.

Read more: Carrying Out Nested Monte Carlo Simulation Using a SubModel (page 992); Viewing SubModel Results in Time History Result Elements (page 617).

Several points should be noted regarding saving and viewing results inside a SubModel:

• When carrying out distributed processing simulations (using the Distributed Processing Module), results are not saved inside a SubModel, even if you choose to Save Results for most recent Simulation.

Note: The Distributed Processing Module is discussed in detail in the GoldSim Distributed Processing Module User’s Guide.

• If the Save Results for most recent Simulation box is cleared, results will not be saved inside the SubModel, even if the SubModel contains Result elements and/or the Save Results options are selected for each element. (If you click on a Result element in Result Mode inside such a SubModel, it will be ignored).

• If 1) you have chosen to Save Results for most recent Simulation, 2) have added a Time History Result element; and 3) the element is specified to automatically export results to a spreadsheet or text file, GoldSim will not automatically export the results. If you wish to export the results of the last simulation from inside a SubModel, it can only be done manually.

Read more: Exporting from a Time History Result Element to a Spreadsheet (page 720); Exporting from a Time History Result Element to a Text File (page 727).

Many models have uncertain (epistemic) parameters as well as randomly variable (aleatory) parameters. An uncertain parameter represents ignorance that can theoretically (but perhaps not practically) be reduced through investigation (e.g., the mean failure time for a batch of light bulbs). Variability is inherent in many systems (e.g., the distribution of failure times for a batch of light bulbs) and cannot be reduced. It is often valuable to explicitly separate variability from uncertainty in a model.
With SubModels, this is accomplished by inserting a SubModel (e.g., a dynamic Monte Carlo simulation) within an outer model (e.g., a static Monte Carlo simulation). This is referred to as "nested" Monte Carlo simulation. The inner model simulates the random variability of the system, while the outer model represents the ignorance in key parameters.

In the example above, the outer model would sample a probability distribution that represents the uncertainty in the mean lifetime of a light bulb, and the inner model would simulate the performance of a number of random light bulbs (whose lifetime is sampled from a distribution with the mean specified by the outer model).

Note that for this kind of model, if you were to link an output from the SubModel to the Output Interface as a distribution, conceptually on the outside of the SubModel the output would actually represent a distribution of distributions. That is, because the SubModel is a Monte Carlo model, every time it is called, it produces a distribution. However, because the outer model itself is a Monte Carlo simulation, every element inside it results in a distribution of results. Hence, running the outer model multiple times results in a distribution of distributions for any SubModel output.

If you were to go inside the SubModel and view a result, it would display only the results from the last time the SubModel was run (and only if Save Results for most recent Simulation was checked).

However, on the outside of the SubModel, for a nested Monte Carlo run, it is necessary to display the distribution of distributions. Of course, GoldSim’s standard Distribution Result only displays a single distribution. (It can display distributions from multiple outputs, but this is not the same as displaying nested distributions).

Read more: Viewing Distribution Results (page 639).

To support nested Monte Carlo simulations, Distribution Result elements modify their behavior under certain circumstances to enable display of nested Monte Carlo results. In particular, Distribution Result elements modify their behavior if and only if the following conditions are all met:

- The Distribution Result element is located in the outer model (i.e., not in a SubModel);
- A single result is selected for display in the Result element;
- The selected result references a distribution type output from a Monte Carlo SubModel (for this to occur, the SubModel must be configured to run multiple realizations);
- The result was added to the Result element prior to running the model;
- The SubModel whose result is being referenced is located within the main model (i.e., it cannot be nested inside another SubModel); and
- The outer model is configured to run multiple realizations.

Under these circumstances, there are two differences from a standard Distribution display:

1. There is no option to display a Table; and
2. A Display option is provided. The options are:
   - All Realizations
   - Realization
The various displays (and what they represent) are as follows:

**All Realizations.** “Realizations” refers to realizations of the outer model. Since each realization of the outer model produces a distribution, showing “All Realizations” simply displays multiple distributions (one for each realization of the outer model). The chart display looks like this:

![Distribution Chart](image)

The Distribution Summary view looks like this:

![Distribution Summary](image)

Statistics can only be shown for one realization at a time (which you can select from a drop list at the top of the dialog). In the example above, statistics are being shown for realization #24 of the outer model.
Realization. “Realization” refers to the realization of the outer model (which you must select). The chart (and Distribution Summary) then show that single outer model realization (i.e., a single distribution):

Statistic. When you select this option, a drop-list to the immediate right allows you to select or define a statistic. The result displayed is then the distribution of the specified statistic (each outer model realization produces one value for this statistic):

Probability. This option displays a probability distribution of probabilities. When you select this option, a drop-list to the immediate right allows you to define the probability that you are interested in, based on a condition:
This instructs GoldSim to compute the (inner model) probability of a particular outcome for the specified result. Each outer model realization produces one value for this probability. It is the probability of the inner model’s result being <, <=, =, >= or > the specified value. In the example above, for each outer model realization, GoldSim computes a single value: the probability of the output being less than 9000 hours. The overall result then displays a distribution that quantifies the uncertainty in this particular probability:

How do we interpret such a distribution? This is meant to represent a complex concept (a probability distribution of a probability!). Perhaps this is best understood by examining this particular example. In this example, we could state the following:

- There is approximately a 70% chance that the probability of the result being less than 9000 hours will be less than one. That is, in 70% of the outer model realizations, the probability of the result being less than 9000 hours was less than one, and in 30% of the realizations it was exactly one.

- There is no chance that the probability of the result being less than 9000 hours is below (approximately) 48%. That is, in none of the outer model realizations was the probability of the result being less than 9000 hours less than (approximately) 48%.

All Results. This combines all realizations together to form a single distribution that represents both variability (from the inner SubModel) and uncertainty (from the outer model). Hence, if you run 50 realizations of the SubModel and 100 realizations of the outer model, the distribution would be constructed from 5000 realizations. The chart (and Distribution
Summary) then show the single distribution that represents all model uncertainties:

![Distribution Graph]

The text above describes how *distribution results* can be viewed for a nested Monte Carlo simulation. But what if you wanted to view time history results? Viewing time history results for nested Monte Carlo simulations is complicated by the fact that Time History Definition SubModel outputs consist of *multiple* time histories for each realization of the parent model. As a result, if the parent model was run for 100 realizations, and the SubModel was run for 50 realizations, the corresponding time history results would be based on 5000 individual time histories (50 * 100).

*Read more:* [Viewing SubModel Results in Time History Result Elements](#) (page 617).

To support nested Monte Carlo simulations, Time History Result elements modify their behavior under certain circumstances to enable display of nested Monte Carlo results. In particular, Time History Result elements modify their behavior to display such results if and only if the following conditions are all met:

- The Time History Result element is located in the outer (parent) model (i.e., not in a SubModel);
- Within the Result Properties dialog, “SubModel Time” is selected for the **Time Display Setting**.
- A single result is selected for display in the Result element. The selected result must reference a Time History Definition output from a SubModel. The **Output Definition** for the output must be “Realization Histories” (for this to occur, the SubModel must be configured to run multiple realizations);

*Read more:* [Creating the Output Interface to a SubModel](#) (page 984).

- The SubModel whose result is being referenced is located within the main model (i.e., it cannot be nested inside another SubModel); and
- The outer (parent) model is configured to run multiple realizations.
Under these circumstances, special **Display** options are provided at the top of the Time History display dialogs:

- Histories for Realization
- Statistic for each Realization
- Histories for all Realizations

The various displays (and what they represent) are as follows:

**Histories for Realization.** “Realization” refers to the realization of the parent model (which you must select). Since each realization of the parent model produces multiple histories for the SubModel, showing “Histories for Realization” simply displays multiple SubModel histories for the selected parent model realization. The chart display looks like this:

Note that the display also provides a separate field (**Show**). In the figure above, the SubModel histories for the specified parent realization (in this case, #8) are being displayed “Individually”. The tool-tip indicates that the particular result indicated by the cursor corresponds to SubModel history #5.

A second option allows these 5 histories to be displayed “As Probabilities” (i.e., percentile ranges):
Read more: Viewing Probability Histories for Multiple Realizations (page 600).

Statistic for each Realization. “Realization” refers to the realization of the parent model. Since each realization of the parent model produces a single statistic (which you must specify) for the SubModel histories, showing “Statistic for each Realization” displays a single time history for each realization of the parent model. Each individual realization represents a statistic computed using the SubModel histories. The chart display looks like this:

In the figure above, there are 10 realizations shown (one for each parent realization). Each realization represents the 50th percentile computed using the 5 SubModel histories that were produced for that parent realization. In this case, each parent realization is being displayed “Individually”. Note that the display also provides a separate field (Show). A second option allows these 10 parent realizations to be displayed “As Probabilities”:

Histories for all Realizations. Since each realization of the parent model produces multiple histories for the SubModel, showing “Histories for all Realizations” simply displays all SubModel histories for all parent realizations. The chart display looks like this:
In the figure above, there are 50 histories shown (5 SubModel histories for each of 10 realizations of the parent model). In this case, each history is being displayed “Individually”. Note that the display also provides a separate field (Show). A second option allows these 50 histories to be displayed “As Probabilities”:

Note: If you have applied screening to the inner model (the SubModel), the screening is ignored for nested Monte Carlo runs. However, if you have applied screening and/or categories to the parent model, the screening is used to create the displays.

Read more: Classifying and Screening Realizations (page 577).

Note: You cannot export time history results (i.e., to a spreadsheet or text file) from a SubModel.

An example illustrating how nested Monte Carlo simulation is carried out (SubModel2.gsm) can be found in the General Examples/SubModel folder in
Using SubModels to Embed Models Within Models

your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

Read more: SubModel Example: Explicit Separation of Variability from Uncertainty (page 1012).

Using Resources Inside a SubModel

Like standalone models and Containers, a SubModel can provide Resources to elements inside it.

Read more: Using Resources (page 839).

To provide Resources to elements inside the SubModel, check the Provide Resources checkbox on the SubModel's Definition tab:

When you do so, a Resources tab is added to the SubModel’s dialog:

This tab can be used to create Resource Types (and global stores) for use inside the SubModel.

Several points should be noted regarding the use of Resources inside SubModels:

- Elements inside a SubModel have no access to Resource Types or Resource Stores outside of the SubModel. Resources in the SubModel are completely separate and independent from Resources in the parent model.

- If you create a SubModel by importing a GoldSim model, and that model has Resources defined, the Provide Resources checkbox will automatically be checked, and any Resource Types and Stores defined in the original model will become part of the SubModel.
Using SubModels to Embed Models Within Models

Read more: Importing SubModels (page 1007).

- If you export a SubModel, any Resource Types and Stores in the SubModel will become available in the exported standalone model.

Read more: Exporting SubModels (page 1006).

The Graphics tab of the SubModel dialog provides access to options for controlling the appearance of elements inside the SubModel:

The top portion of the dialog (the Graphics Pane section) is used to define the size of the graphics pane, a background color for the graphics pane, and whether or not a grid is displayed.

Read more: Adjusting the Size of the Graphics Pane (page 425); The Graphics Pane Grid and Background (page 423).

The bottom portion of the dialog (the Influences section) is used to specify the default shapes for influences in the SubModel, and to define whether and in what manner influences within the SubModel are filtered (hidden). The options in this portion of the dialog are quite important, as they can be used to ensure that your models are easy to view and understand.

Read more: Links and Influences (page 80); Editing the Appearance of Influences (page 426); Filtering Influences (page 430).

The Information tab of the SubModel dialog displays some summary information regarding the SubModel:

Controlling the Appearance of the Graphics Pane for a SubModel

Viewing and Editing SubModel Summary Information
Using SubModels to Embed Models Within Models

The top part of the dialog provides two fields (Author and Analysis description) which allow you to identify the author and provide a brief description for your SubModel.

The bottom part of the dialog provides some useful summary statistics for the SubModel, including the number of model elements and graphical components, and the degree of subcontainment. The number of levels of subcontainment does not refer to the number of Containers; it refers to the number of hierarchical levels of Containers. For example, if the SubModel A contained Containers B and C, A would have 1 level of subcontainment; if the SubModel contained B, and B contained C, it would have 2 levels of subcontainment.

The dialog also indicates the total size of the results being saved for all elements within the SubModel. This information can be very useful in helping you to manage the size of the model file.

One of the applications for a SubModel is to carry out a dynamic optimization (i.e., at specified times) during a simulation. For example, imagine a situation where you were simulating the operation of a facility over a period of one year. Every month, the operators make a decision based on the current state of the system. This decision is based on a simple optimization analysis using currently available data (i.e., at every simulated month during the simulation). The optimization chooses the optimum values of a few control variables that they will use for the next month.

To represent this in GoldSim, you would need to specify the Solution Type for the SubModel as "Optimization", and the SubModel itself would represent the optimization calculations carried out by the simulated operator.
Note: If a SubModel is set to carry out an optimization, it cannot contain any Spreadsheet elements.

When you select "Optimization" for the Solution Type, GoldSim enables optimization for the SubModel and immediately switches to the Optimization tab for the SubModel:

This is the standard optimization dialog that is also used for stand-alone models, and is described elsewhere in detail.

Read more: Running an Optimization (page 528).

Several points regarding use of this dialog for SubModel optimization, however, are worth noting:

- Optimization Variables can be defined in terms of variables that are outside of the SubModel. That is, initial values, upper bounds and lower bounds for Optimization Variables can be defined in terms of a variable that is placed on the input interface to the SubModel.

- In most cases, all of the Optimization Variables themselves (and perhaps the Objective Function) should be placed on the output interface to the SubModel (since the purpose of the SubModel is typically to determine these variables). When running an Optimization in a SubModel, the only applicable result type is Final Value. At the end of the optimization, the SubModel outputs the final (optimized) values of the Optimization Variables.

- Optimizations can sometimes fail or may be unable to converge. GoldSim provides different types of error messages when carrying out optimizations. If an optimization of a SubModel fails, a fatal warning
message will be displayed (and the simulation is halted). If the optimization completes, but cannot fully converge, a message will be written to the run log (but the simulation will not be halted).

Read more: Understanding Optimization Warning Messages (page 537).

**Protecting the Contents of a SubModel**

In some cases, you may want to protect the contents of your SubModel by sealing or locking it. SubModels can be sealed and locked in the same manner as Containers.

Read more: Sealing and Locking Containers (page 143).

To access the sealing/locking dialog for a SubModel, check the Protect SubModel Contents from Changes checkbox in the SubModel dialog. When you do so, the Sealing/Locking dialog will be displayed:

After you seal or lock the SubModel, the protected status of the SubModel is displayed in the SubModel dialog:

**Controlling How Run Messages are Logged for a SubModel**

When using SubModels, you can control whether or not Run Log messages related to the SubModel are written to the parent model’s Run Log.

Read more: The Run Log (page 548).

This can be controlled from within the SubModel dialog:
By default, **Logging of SubModel Run Messages** is set to "Include". Optionally, however, you can set this to "Ignore".

When SubModel messages are sent to the Run Log, it is clearly indicated which model or SubModel generates the message.

**Exporting SubModels**

In some cases, you may want to run a SubModel as a standalone model. This is useful, for example, if you want to debug, test, and/or refine it separate from the parent model. Although you can't run the SubModel directly from within a parent model, you can export it, run it separately, and then import it back into the parent model. You may also choose to export a SubModel so that you can subsequently import it into a different model.

*Read more:* [Importing SubModels](#) (page 1007).

You can export a SubModel by pressing the **Export as .GSM…** button in the SubModel dialog. When you do so, you will be prompted for a file name (and location) for the saved file.

When exporting a SubModel, several points should be noted:

- All of the options (specified in the parent model's Options dialog) are exported with the new model.
- All user-defined units and array label sets specified in the parent model are exported with the new model.
- If you export a SubModel, any Resource Types and Stores in the SubModel will become available in the new model.

*Read more:* [Using Resources Inside a SubModel](#) (page 1001).

In addition, the input and output interfaces are exported to the stand-alone model. The output interface is not accessible in the stand-alone model, but the input interface is accessible, and can be edited. In particular, the input interface is accessible via the **Globals** tab in the stand-alone model's Simulation Settings dialog:
Using SubModels to Embed Models Within Models

Read more: Defining and Referencing Global Properties (page 484).

In the example shown above, Parameter1 and Parameter2 in the SubModel were referencing outputs X and Y in the parent model. Once the SubModel was exported, of course, it no longer had access to X and Y "outside" of the model (since there is no "outside" to a stand-alone model). Hence if you tried to run the stand-alone model an error would be displayed.

In order to run the stand-alone model, the Definitions for the Globals would need to be changed to values. Note, however, that the original exported Definitions are saved with the file separately. Therefore, if you subsequently import the stand-alone model into another model (as a SubModel), the original input (and output) interface that was exported will be automatically recreated.

Note: You can only export a SubModel that has values or conditions on the input interface. If the input interface contains Time Series Definitions, Lookup Table Definitions or Distribution Definitions, you will not be able to export the model (as these could not be represented as values on the Globals tab.

If the exported SubModel's input interface included externally defined Run Properties (e.g., such as Simulation Duration), these interface items will appear in the Globals list. However, they will not be used by the stand-alone model, and cannot be edited. They can, however, be deleted.

Importing SubModels

In some cases, you may want to import an existing stand-alone model into another file as a SubModel. This is useful, for example, if you want to export a SubModel to test (and refine it) outside of the parent model, and then import it back into the parent model.

Read more: Exporting SubModels (page 1006).
To import a stand-alone model into your current model as a SubModel, you must insert a SubModel from the Insert Element menu. When you insert a SubModel, you are presented with a dialog asking if you want to create a new (empty) SubModel, or create the SubModel by importing an existing standalone model. You should choose the latter. When you do so, you will be prompted for a filename and location. After selecting it, the specified file will be inserted into the parent model as a SubModel.

There are several rules that must be met in order for the model to be successfully inserted as a SubModel:

- The stand-alone model must have been saved using the same version of GoldSim that is trying to import it.
- The stand-alone model must be in Edit Mode.

**Note:** If the model being imported contains Scenarios, the scenario information will be removed when it is imported and only the information pertaining to the Active Scenario will be imported.

---

When importing a model, all of the information on the Globals tab of the Simulation Settings dialog in the stand-alone model is imported into the input interface for the SubModel.

**Read more:** [Creating, Running and Comparing Scenarios](#) (page 504).

---

**Note:** If the model being imported was previously exported from a SubModel to a stand-alone model, the original exported Definitions for the input and output interface will be automatically recreated when the model is imported to recreate it as a SubModel. Any changes to existing input Definitions will be ignored (the Definitions that existed when the model was exported will be used). Any new input Definitions that were added after the model was exported, however, will be added to the interface.

---

If you import a GoldSim model as a SubModel, and that model has Resources defined, the **Provide Resources** checkbox will automatically be checked, and any Resource Types and Stores defined in the original model will become part of the SubModel. The Resources in the SubModel and those in the parent model are completely separate and independent. Elements in the SubModel cannot see or interact with Resources in the parent model.

**Read more:** [Using Resources Inside a SubModel](#) (page 1001).

Because the parent model and the imported model potentially could have conflicting definitions (for units, array label sets and version stamps), the following rules are followed during the import to ensure consistency.

**Ensuring Consistency of Array Label Sets**

When importing an existing model as a SubModel, all of its array label sets are imported into the parent model. Therefore, existing sets and imported sets must be merged. The following rules apply when importing array label sets:

- Any set that exists in the imported model but not in the parent model is automatically added to the parent model.
Using SubModels to Embed Models Within Models

- If the imported model and the parent model have sets with identical names (set names are not case-sensitive) that are not otherwise identical (i.e., they must have the same number of items with the same index names), an error is displayed and the import of the model will fail.

Read more: Understanding Array Labels (page 783).

Ensuring Consistency of Units

When importing an existing model as a SubModel, all available user-defined units in the existing model are imported into the parent model. Therefore, existing units and imported units must be merged. The following rules apply when importing units:

- If the imported unit name is unique, it is added to the parent's unit list.
- If the imported unit name is identical to an existing unit name, and the units have the same dimensions, but are assigned to different categories, then the unit is assigned to the category specified by the parent model.
- If the imported unit name is identical to an existing unit name, and the units have the same dimensions, but are assigned different conversion factors, then the conversion factor is assigned the value specified by the parent model.
- If the imported unit name is identical to an existing unit name, but the units have different dimensions, an error is displayed and the import of the model will fail.

Read more: Creating New Units (page 441).

Ensuring Consistency of Versioning

When importing a model file that includes versioning information, GoldSim evaluates the version stamps and compares them against the version stamps in the existing model (if it uses versioning).

If all version stamps are identical (i.e., all properties must match: stamp number, name, description, user name), then the imported model elements keep their individual version logging information. Globally logged information from the imported model, however, is lost.

If the version stamps are not identical, or if the existing (parent) model does not use versioning, all version information is removed from the imported model and its elements.

Read more: Tracking Model Changes (page 1023).

Interrupting and Pausing a Simulation Within a SubModel

When you pause a model using the Run Control toolbar or via an Interrupt element message, if the model was in the middle of computing an element within a SubModel, GoldSim will pause the model at that point.

Read more: Pausing and Stepping through a Simulation (page 502); Continuing, Pausing and Aborting a Simulation After an Interrupt (page 412).

Note, however, that when a model is paused, the status bar always displays the status of the outermost parent model.

Viewing Element Dependencies Within a SubModel

In complex models, it is often useful to explore the interdependencies of the various elements (i.e., who affects who). GoldSim provides two very powerful utilities for doing this: the Function Of View, and the Affects View. If you right-click on an element in either the graphics pane or the browser (to access the
context menu) and select **Function Of...** or **Affects...**, a floating browser window is displayed that shows the element dependencies in the form of a tree:

![](image)

**Read more:** [Viewing Element Dependencies](page 112).

If the tree includes a SubModel, the SubModel is effectively treated as an element (this is in contrast to a Container, which generally would not show up in the tree at all). That is, the SubModel is treated as any other element with inputs and outputs would be treated. With regard to the dependency tree, it is simply a “custom” element. In the example below, the Expression A is shown to be a function of Y and the SubModel:

![](image)

In many instances, however, you may be interested in looking inside the SubModel to view the dependencies. That is, it may not be enough to know that A is a function of SubModel1; you may want to explore exactly what A is a function of within the SubModel.

To facilitate this, GoldSim allows you to expand SubModels within a Function Of or Affects tree:
When GoldSim displays items inside a SubModel in this way, it always does the following:

- The SubModel is clearly labeled as such.
- The contents of the SubModel in the tree are shaded (orange) to indicate that these items are inside the SubModel. (If there are SubModels within SubModels, different shades are used to indicate this).
- In addition to showing the elements inside the SubModel, GoldSim also displays any items on the input and output interfaces that may be involved (and these are clearly labeled as such). In the example above, the tree is indicating that A is a function of the “Prob” output on the SubModel interface, which is a function of the element W inside the SubModel, which is a function of the “~X” input on the SubModel interface, which is a function of the X element outside of the SubModel.

**SubModel Examples**

Several example models illustrating the use of SubModels are described below. These examples can be found in the General Examples/SubModel folder in your GoldSim directory.

These examples cover the four most common applications for SubModels.

**Read more:** What Can I Do With a SubModel? (page 973).

**SubModel Example: Manipulation of Monte Carlo Statistics**

A common use of SubModels is to manipulate Monte Carlo statistics. That is, after carrying out a Monte Carlo simulation, you may want to carry out further calculations using the statistical outputs of the simulation. For example, you may want to carry out a calculation that is some function of the mean and the 95th percentile of a particular output in a Monte Carlo simulation.
Using SubModels to Embed Models Within Models

Without the use of SubModels, the only way to accomplish this is by exporting results from a Monte Carlo simulation manually to another application (e.g., a spreadsheet or a separate GoldSim model). With SubModels, this is easily accomplished within a single GoldSim model by inserting a SubModel (that is a Monte Carlo simulation) into an outer model (that is static and simply manipulates the statistical outputs of the inner model).

Example model MonteCarloStatistics.gsm in the General Examples/SubModel folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu) provides a simple illustration of such an application. In this model, the SubModel contains a single Stochastic. The SubModel carries out a Monte Carlo simulation. The outer model is a static deterministic simulation that contains a single Expression element. The Expression is a function of statistics (the mean and 95th percentile) of the Stochastic within the SubModel.

Another common use of SubModels is to explicitly separate variability from uncertainty. That is, many models have uncertain parameters as well as variable parameters. An uncertain parameter represents ignorance that can theoretically (but perhaps not practically) be reduced through investigation (e.g., the mean failure time for a batch of light bulbs). Variability is inherent in the system (e.g., the distribution of failure times for a batch of light bulbs) and cannot be reduced.

It is often valuable to explicitly separate uncertainty from variability in a model. With SubModels, this is accomplished by inserting a "variability" SubModel (e.g., a dynamic Monte Carlo simulation) within an outer "uncertainty" model (e.g., a static Monte Carlo simulation). This is referred to as nested Monte Carlo simulation.

Read more: Carrying Out Nested Monte Carlo Simulation Using a SubModel (page 992).

Example model UncertaintyVariability.gsm in the General Examples/SubModel folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu) provides a simple illustration of such an application. In this model, a nested Monte Carlo simulation is used to calculate the confidence in the performance of a system of light bulbs. The outer model samples two probability distributions for values that describe the shape of the failure distribution for the light bulb. The fact that these are distributions indicates that we are uncertain about the precise shape of the failure distribution. The inner model then simulates the performance of the light bulbs (whose lifetime is sampled from a failure distribution with the (uncertain) variables provided by the outer model). The result of this type of analysis is a probability of probabilities (e.g., what is the chance that the probability of the light bulbs lasting less than 6000 hours will be less than one?).

Another common use of SubModels is to optimize a probabilistic model. If you wish to optimize a probabilistic (uncertain) system, the objective function to be optimized cannot be a single deterministic output. Rather, it must be a statistic. That is, if X was an output of a probabilistic model (and hence was output as a probability distribution), optimizing X itself would be meaningless. Rather, you would need to optimize a particular statistic (e.g., the mean or 50th percentile) of the output X.

With SubModels, this is accomplished by inserting a SubModel (e.g., a dynamic Monte Carlo simulation) within an outer model (e.g., a static optimization).

Example model ProbabilisticOptimization.gsm in the General Examples/SubModel folder in your GoldSim directory (accessed by selecting
Customized Importance Sampling Using User-Defined Realization Weights

File | Open Example... from the main menu) provides a simple illustration of such an application. In this model, the SubModel is a Monte Carlo simulation containing a single Stochastic. The outer model is an optimization. The variable being optimized is a function of another element in the outer model and a statistic (the $95^{th}$ percentile) of the Stochastic in the SubModel.

Another common use of SubModels is to do a dynamic optimization during a simulation. Imagine a situation where you were simulating the operation of a facility over a period of one year. Every month, the operators make a decision based on the current state of the system. This decision is based on a simple optimization analysis using currently available data (i.e., at every month during the simulation). The optimization chooses the optimum values of a few control variables that the operators will use for the next month.

With SubModels, you could simulate this by inserting a SubModel (e.g., a static optimization) within an outer model (e.g., a dynamic simulation).

Example model DynamicOptimization.gsm in the General Examples/SubModel folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu) provides a simple illustration of such an application. In this model, the outer model is a dynamic deterministic simulation, and the SubModel is a static deterministic optimization. The objective function in the SubModel is a function of a parameter that is passed in from the outer model (and changes every timestep). Hence, every timestep, the SubModel carries out a new optimization and the results are used in the parent model for the next timestep.

Customized Importance Sampling Using User-Defined Realization Weights

For risk analyses, it is often necessary to evaluate the low-probability, high-consequence portions of the probability distribution of the performance of the system, and that can require carrying out a very large number of Monte Carlo realizations. Because the models for such systems are often complex (and hence need significant computer time to simulate), it can be difficult to use the conventional Monte Carlo for such risk analyses.

To facilitate this type of analysis, GoldSim allows you to utilize an importance sampling algorithm to modify the conventional Monte Carlo approach so that selected portions of input distributions (which could correspond to high-consequence, low-probability outcomes) are sampled with an enhanced frequency. During the analysis of the results that are generated, the biasing effects of the importance sampling are reversed. The result is high-resolution development of “high-consequence, low-probability outcomes”; without paying a high computational price.

Stochastic elements provide built-in functionality to force importance sampling on either the low end or high end of a Stochastic element’s range (i.e., the tails of distribution). Timed Event and Random Choice elements provide similar functionality.

Read more: Applying Importance Sampling to a Stochastic Element (page 181); Timed Event Elements (page 365); Random Choice Elements (page 373).

In some cases, however, it may be necessary apply importance sampling not just over the tails, but over user-defined regions of the Stochastic element’s range. For example, if it was known that the important results of a model were highly
sensitive to values of a particular Stochastic that were close to the 40th percentile, you would want to focus sampling in that region.

GoldSim supports this by providing three specialized functions that provide biased sampling (and compute the appropriate weights) for a targeted range within a probability distribution. All three functions require a Target probability and a Width. For example, if you specify a Target of 0.4 and a Width of 0.05, biased importance sampling will be focused in the region between probability level 0.375 and 0.425. The functions provided by GoldSim are as follows:

**ImpProb(Old,Target,Width):** Takes a sampled probability (Old) and returns a probability biased toward the specified region (the manner in which this is done is discussed below).

**ImpWeight(Prob,Target,Width):** Takes a biased probability (produced by ImpProb) and returns a weight.

**ImpOld(Prob,Target,Width):** Takes a biased probability (produced by ImpProb) and returns the unbiased probability. This is the inverse of ImpProb.

By using these functions in conjunction with GoldSim’s option for the user to manually specify realization weights, a customized importance sampling scheme can be implemented.

The steps required to implement importance sampling on a single Stochastic element (e.g., named Stoch1) with a Target of 0.4 and a Width of 0.05 are as follows:

1. Create a Stochastic element (e.g., U) defined as a Uniform between 0 and 1. This will generate a random number.

2. Create an Expression element (e.g., Prob) defined as “ImpProb(U,0.4,0.05)”. This will generate biased random numbers with which to sample Stoch1.

3. Create an Expression element (e.g., Stoch1Value) with the same dimensions of Stoch1, defined as “PDF_Value(Stoch1.Distribution,Prob)”. This will generate the biased sample of Stoch1.

   **Read more:** Specialized Functions that Operate on Distributions (page 182).

4. Create an Expression element (e.g., Weight) defined as “ImpWeight(Prob,0.4,0.05)”. This computes the appropriate weight to use for the realization.

5. Open the Simulation Settings dialog, and go to the Monte Carlo tab. In the Probabilistic Simulation section, check the Specify Realization Weights box. An Insert link dialog will appear, and you should select the Weight element defined in the previous step. This then instructs GoldSim how to appropriately weight each realization when computing results (e.g., CDFs).

   **Read more:** Probabilistic Simulation Options (page 479).

This specific example can be found in the file CustomImportance.gsm in the General Examples/Stochastic folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

How does GoldSim actually implement this importance sampling? The implementation is as follows:
During the Monte Carlo simulation half of all the random numbers generated by the importance sampling function (ImpProb) will lie outside of the enhanced-sampling region (defined as the region from Target – Width to Target + Width), and half inside it. The ‘outside’ and ‘inside’ regions are sampled differently:

- The half of the samples that are outside the region are distributed uniformly.
- The sampling of the half that are inside of the zone depends on whether the target probability is close to 0 or 1:
  - Normally, these samples have increased sampling rates closer to the target value. The sampling rate within the enhanced sampling zone is distributed according to a truncated normal distribution, with a standard deviation equal to one quarter of the width of the zone.
  - However, if the target probability level is less than the specified width, or greater than (1 – the specified width), then the enhanced samples are also uniformly distributed, though sampled at a higher rate than the samples outside of the region.

The figure below shows how a set of random numbers (generated by sampling from a uniform distribution between 0 and 1) is modified (biased) using the ImpProb function to focus on the specified Target. An unbiased uniform distribution simply produces a CDF that is a straight line with a slope of 1. The biased distribution of random numbers (in this case with a Target of 0.4 and a Width of 0.05 focuses 50% of the samples to fall between 0.375 and 0.425. The remaining 50% falls outside this range (in this case 20% below and 30% above).

Of course, the weights of these samples are adjusted accordingly (using the ImpWeight function) such that in this case, the 50% of the samples falling between 0.375 and 0.425 account for only 10% of the total weight.

The following points should be noted regarding customized importance sampling:

- The Target must be defined as a value between 0 and 1.
- The Width must be between 1E-6 and 0.5.
Dynamically Revising Distributions Using Simulated Bayesian Updating

GoldSim provides an advanced feature for dynamically revising distributions during a simulation using “simulated Bayesian updating”. This feature is most valuable when using GoldSim to simulate a project of some kind (e.g., developing a new product, building a new facility). It is often the case that as a project progresses, additional information about uncertain variables is acquired. From the project operator’s point of view, their uncertainty is decreasing, and hence as time progresses, the probability distributions used in their model of the project are converging towards the true values of the variables. As their knowledge increases they are better able to make appropriate decisions for the project’s path forward.

When simulating a project in this manner using GoldSim, the model that is constructed will use Stochastic elements to specify the initial probability distributions of uncertain parameters, and a Monte Carlo sampling process will be used to pick the ‘true’ values of those parameters. Given that starting point, simulated Bayesian updating can be used to simulate the additional information that will accrue as the project progresses. This in turn can be used to simulate the decisions that would be made based on the new information.

To support this, GoldSim allows you to define an “uncertainty reduction factor” for an uncertain variable. To do so, you must create a new Stochastic and specify its distribution type as an “Externally-defined Distribution”:

- Using the approach outlined above, any correlation factor specified for Stoch1 will be ignored (since the value is not generated by the distribution, but via the PDF_Value function).

- This approach can readily be applied to do enhanced sampling of multiple Stochastic elements. It is simply necessary to 1) use a different (uncorrelated) random number (U in the example above) for each element; and 2) multiply all of the independent realization weights together to generate the Realization Weight used in the Simulation Settings dialog. However, when multiple elements use importance sampling it is important not to use narrow sampling regions, as this can lead to numerous ‘wasted’ realizations that have negligible realization weights.

- The custom importance sampling approach is compatible with the built-in functionality to force importance sampling on either the low end or high end of a Stochastic element’s range. Hence, your model could contain some elements that use the “built-in” approach and others that use the custom approach.
Dynamically Revising Distributions Using Simulated Bayesian Updating

The uncertain variable (whose uncertainty is being reduced as the model progresses) is entered in the External Definition field for this new Stochastic. This must be a link to the Distribution output of a Stochastic element. The Distribution output is a complex output that contains all the information regarding the distribution. It is the fourth output of a Stochastic:

Read more: Externally-Defined Distribution (page 167).

You must then check the Use Bayesian Updating checkbox, and then enter a value for the Uncertainty Reduction Factor. This dimensionless factor, which must be between 0 and 1, approximates the extent to which the original distribution’s standard deviation has been reduced at any point in time. Essentially, this allows you to define how the initial uncertainty about the original distribution is reduced. As such, the Uncertainty Reduction Factor will almost always have an initial value of 1 (no reduction of uncertainty), and become smaller as a function of time (typically by changing in discrete steps representing points in time when new information has been obtained).

To understand the mechanics of how simulated Bayesian updating works, let us first examine a simple model that illustrates how the Externally-defined distribution is modified by a time-variable Uncertainty Reduction Factor. Subsequently, we will describe a (simplistic) real-world application.
The example model described here can be found in the file BayesianUpdating.gsm in the General Examples/Stochastic folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu). It is recommended that you open that file and follow along during the discussion below.

In a Container named “Basic_Concepts”, you will see a very simple model that looks like this:

The True_Ore_Grade represents the variable that is being studied (and whose uncertainty is being reduced during the simulated project). In this example, it is a Normal distribution with a mean of 10 and a standard deviation of 3. The Estimated_Ore_Grade represents the updated distribution resulting from studies carried out on the ore grade during the project. It is an External Distribution whose primary input is the True_Ore_Grade, and whose Uncertainty Reduction Factor is defined by the variable URF:

URF is simply an Integrator with an initial value of 1 (no uncertainty reduction) that is updated by a Discrete Change at discrete points in time (representing studies that have reduced the uncertainty in the ore grade). In this example, URF is simply reduced by a factor of 0.2 every 10 days for a period of 50 days:
Hence, after 5 days, the uncertainty in the ore grade has actually been reduced to zero (i.e., it is assumed that the ore grade is perfectly known at 50 days).

Given this information, it is instructive to view how the Estimated_Ore_Grade distribution changes as a function of time. Although we can’t easily view the distribution in the middle of a simulation, we can examine its statistics and see how they change with time. To do so, we take advantage of two of GoldSim’s built-in functions: PDF_Mean and PDF_SD, which return the mean and standard deviation, respectively, of a specified distribution.

Read more: Specialized Functions that Operate on Distributions (page 182).

The plot below shows how the mean and the standard deviation of the Estimated_Ore_Grade distribution change with time. The plot also includes the True_Ore_Grade:

Every realization of the model produces a different sampled True_Ore_Grade. In this particular realization, the True_Ore_Grade was sampled to be just less
than 8. As can be seen, the initial mean and standard deviation of the Estimated_Ore_Grade are 10 and 3, respectively (the same as the statistics for the True_Ore_Grade distribution). However, starting at 10 days, more information is learned about the ore grade. This has two effects on the Estimated_Ore_Grade distribution: 1) the standard deviation is reduced; and 2) the mean approaches the true value. Whenever new information is obtained (in this example, every 10 days), the standard deviation is further reduced, and the mean moves closer to the true value. At 50 days (when it is assumed that “perfect information” is obtained), the standard deviation goes to zero, and the mean converges to the true value.

Now that we have explained conceptually how simulated Bayesian updating is carried out, let’s now expand on this example slightly to illustrate how it might actually be used in a real model. The example model described here can be found in the same file (BayesianUpdation.gs) in the General Examples/Stochastic folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu). It is in a Container named “Example_Application”.

The example considers the case of a proposed new mine. Initially the mean ore grade may have considerable uncertainty, but as successive studies are carried out, additional information is obtained and the uncertainty about the mean ore grade is reduced. Let’s further assume that once you have a particular level of certainty regarding the value, you will act in one way or another. We will represent this by starting with the previous example, and making one change and one addition:

- We will assume that only four studies are carried out (at 10, 20, 30 and 40 days), each reducing the Uncertainty Reduction Factor by 0.2, such that (as would be realistic) perfect information regarding the ore grade is never obtained.

- We will assume that the estimated ore grade must be greater than 12 for the project to proceed. Because the ore grade cannot be known with certainty, however, this decision must be based on probabilities. In particular, when simulating the project, it is assumed that the project will proceed if and only if we have at least 50% confidence that the ore grade is greater than 12 (i.e., we will proceed if the probability of the ore grade being at least 12 exceeds 50%).

The probability of the ore grade being at least 12 is another statistic (that is changing with time as more information is obtained) that can be computed using one of GoldSim’s built-in functions (PDF_CumProb). The plot below shows how the probability of the ore grade being at least 12 changes with time for a particular realization:
In this realization, the true value is sampled to be approximately 14.3. Given the original distribution (Normal distribution with mean 10 and standard deviation 3), the original probability of the value being greater than 12 is about 25%. As the uncertainty in the ore grade is reduced, the probability of the ore grade being greater than 12 increases, until at 30 days, it jumps above 50% to 57% (and at 40 days jumps to 93%), and hence the project can proceed.

In this simplified model, when this occurs, an event is triggered which in a real model could trigger the next phase to begin:

Although this example is simple, it illustrates how simulated Bayesian updating can be used to simulate additional information that is obtained as a project progresses, and how this can in turn be used to simulate the decisions that would be made based on the new information.

The conventional approach to sampling an uncertain variable is to first sample a random value (from a uniform distribution between 0 and 1), and then use this value as input to the inverse cumulative distribution function for the uncertain variable.

For example, the CDF for a Normal distribution (with mean 10 and standard deviation 3) looks like this:
To sample this distribution, GoldSim would first sample a random value from a uniform distribution between 0 and 1 (such that all values were equally likely), and use this as the cumulative probability level for the sampled variable. It would then map that probability to a value. For example, if 0.4 was sampled from the uniform distribution, that would map to a value for the variable of approximately 9 in this case.

To understand how simulated Bayesian updating is carried out, recall that two separate Stochastic elements must be defined to carry it out: 1) a Stochastic representing the “true” value; and 2) a Stochastic (defined as an Externally-defined distribution with a specified Uncertainty Reduction Factor) representing the “updated” distribution. When doing Bayesian updating, the key point is that the random value on the probability axis for the “updated” distribution is not sampled from a uniform between 0 and 1; rather, it is sampled from a beta distribution, whose shape depends on the sampled “true” value and the specified Uncertainty Reduction Factor for the “updated” distribution.

The specific way this is done is as follows:

1. A random number is selected in order to sample the distribution representing the “true” value.

2. In order to sample the “updated” distribution, a beta distribution is created on the probability axis for the distribution. This beta distribution has a minimum value 0 and maximum value 1. Its standard deviation is calculated as the product of the default standard deviation value for sampling the probability axis (0.288675, for a uniform distribution) and the Uncertainty Reduction Factor. The mean of the beta moves linearly from the original mean (0.5 for a uniform distribution) towards the sampled probability level for the original Stochastic as the Uncertainty Reduction Factor ranges from 1 to 0.

Hence, if we assume that the sampled random number for the “true” value is 0.3, and the Uncertainty Reduction Factor is 0.5, then the beta distribution that is created would have a mean of 0.4 (half way between 0.5 and the sampled value 0.3) and a standard deviation of 0.288675*0.5.

3. This beta distribution is then sampled (returning a value between 0 and 1).

4. This value is then used as input to the inverse cumulative distribution function for the original Stochastic.
Note that as the Uncertainty Reduction Factor approaches 0, the “updated” value approaches the “true” value.

The approach described above is used for all continuous distributions. As described above, it effectively increases the likelihoods of “updated” values closer to the “true” result, and reduces the likelihoods of values that are further away from it. This is usually an appropriate approach, as most tests of continuous variables will have a limited accuracy and resolution, and as such, the probabilities of nearby values are correlated. However, for a discrete distribution (e.g. “is it an apple, an orange, or a pear?”) there is likely no correlation between the probabilities of nearby values (and for Boolean, the concept of “nearby” values does not even apply). Accordingly, a different approach (described below) is used for Boolean and Discrete distribution types. However, the approach described above for continuous distributions is also used for the results of binomial, Poisson, and other discrete-valued distributions, as there typically would be a correlation between nearby values for these distributions.

For the Boolean and the Discrete distributions the uncertainty reduction works differently. In these cases, for the “updated” distribution, the probability of the “true” value is increased linearly from its original probability level to a value of 1 as the Uncertainty Reduction Factor drops from 1 to 0. For all other outcomes, the original probabilities are simply multiplied by the Uncertainty Reduction Factor, dropping to 0 as the Factor approaches 0. As was the case for the continuous distributions, sampling is accomplished via an auxiliary probability distribution that modifies the sampled probability levels, but instead of a beta distribution, a cumulative distribution is used in order to have the desired effect.

Tracking Model Changes

GoldSim provides the ability to track changes that you have made to your model file. This feature (referred to as versioning) allows you to quickly determine the differences between the current version of your model file and some previous version of the file.

Providing this configuration management capability is particularly useful for:

- coordinating model changes when multiple people can access and modify the model file;
- as a Quality Assurance/Quality Control feature enabling you to demonstrate and document when and what changes have been made to a model file.

Versioning Overview

Changes to a model file are tracked by creating model file versions. A version is an internal “snapshot” of your model file at a particular point in time. You must tell GoldSim when to take a “snapshot” by creating a version, and assigning it a title (e.g., “1-12-2002”, “Initial Model”, “Revision A”). Once you have created at least one version, you can then compare the current model (the model as it exists right now) to any previous model version you have created.

GoldSim can report the differences between the current model and any previous version. Note that GoldSim does not actually tell you how a model has changed; in general, it only reports what has changed.

For example, if you change some of the inputs to a Reservoir element between versions, GoldSim will report that the element has been changed, but it does not store or report which inputs changed or what their previous values were. Similarly, if you delete an element from a Container, GoldSim will report that an element with a particular ID was deleted from the Container, but it does not
Tracking Model Changes

record the type of element (or the manner in which the element was defined). This keeps the size of the model file (where all of the changes are stored) manageable.

Generally, you will want to create a version whenever some project milestone has been met. The milestone might be the end of a particular phase of the project, or after you have made some major modification to the model. In some cases, you may want to create versions on a regular basis (e.g., every Friday).

Immediately after you create a version, it may be useful to archive a copy of the model file. If you do so, you will then be able to determine the details of the changes that were made between versions.

Read more: Saving, Opening and Closing GoldSim Files (page 62).

For example, when GoldSim reports that element A was modified between version X and the current version, you can open and compare the file that was archived when version X was created to determine how the element was defined at that point. When you archive a copy of your current model, GoldSim records the name of the archived file (along with the version number) in your current model.

Read more: Changes Tracked Between Versions (page 1026).

Enabling Versioning

In order to enable versioning in a file, select Model|Versioning… from the main menu. The first time you do this, a message is displayed asking you to confirm that you want to enable versioning.

After confirming your request, you will immediately be prompted to create a version of the file (take a “snapshot” of the current model file and “stamp” it as a version).

Once you create the first version, versioning will be enabled, and will remain enabled until you choose to disable it.

Read more: Disabling Versioning (page 1026).

Creating Versions

When you first enable versioning, or whenever you choose to create additional versions via the Version Manager, a dialog for creating versions is displayed.

Read more: The Version Manager (page 1025).

You must assign a Version Title to each version. This Title is not required to be unique (although in most cases it should be). It is used in messages and reports when comparing versions, and should be something meaningful and explicit (e.g., “Draft #1”, “Final Model”, “Jim’s Modifications”, “January 12”).

The User Name will also appear in difference reports and messages, and is intended to identify the model user who created the version. By default, the Windows user name will appear, although you can edit this.

The Description field can be used to add descriptive text for each new version.
Finally, you must assign a **Version** number of the form X.yyy (e.g., 1.003). GoldSim ensures that new version numbers are always greater than the previous version number.

**The Version Manager**

Once you have defined your first version, or if you select **Model|Versioning** after versioning has already been enabled, the Version Manager dialog is displayed.

The Version Manager lists all the versions that you have created. If you select one of these versions, the details of that version (user name, date created, and description) are displayed at the bottom of the dialog:

![Version Manager dialog](image)

You can edit the **Description** here, but cannot change any other details about the version.

You can create new versions by pressing the **Create Version…** button, which provides access to the Create Version dialog.

*Read more: Creating Versions* (page 1024).

**Deleting a Version**

You can delete an existing version by selecting it in the Version Manager and pressing the **Delete Version** button. Note, however, that when you delete a version, you also automatically delete all previous versions.

When you delete a version, the version number is never used again. For example, if you had a single version (1.000), and you deleted this version and then re-enabled versioning and created another major version, it would be numbered as 2.000.

If you choose to delete the most recent version (and hence all versions in the model), GoldSim will automatically disable versioning (until you enable it again by creating a new version).

**Specifying the Reference Version**

Once you have created at least one version, you can then compare the current model (the model as it exists right now) to any previous model version you have created. In order to do so, you must identify the **Reference Version** (the version to which the current model is compared).

This is done by selecting the checkbox to the right of the version in the Version Manager. Only one checkbox can be selected at a time (i.e., there can be only one Reference Version).
You can readily change the Reference Version while you are viewing differences.

Read more: Displaying Version Differences (page 1027).

Disabling Versioning

You disable (remove) versioning by pressing the Disable Versioning button in the Version Manager dialog. When you disable versioning, all versioning information is deleted (and cannot be recovered). Hence, before doing so, GoldSim will ask you to confirm that you want to delete all versioning information.

Changes Tracked Between Versions

When versioning is enabled, GoldSim can report the differences between the current model and any previous version. GoldSim does not actually tell you how a model has changed; it only reports what has changed. For example, if you modify an input field of an element, GoldSim does not store what field was edited or what the previous value was. Rather, it simply reports that “one or more of the element’s attributes have been changed”.

The changes that are logged by GoldSim can be divided into two categories: global changes and element-specific changes. Global changes pertain to the entire model. Element-specific changes pertain to a particular element.

The following global changes are recorded and stored within the model file when versioning is enabled:

- A new version is created.
- A version is deleted.
- Versioning is disabled.
- The model filename has changed via a Saved As command (the previous filename is recorded).
- The model filename has been changed outside of GoldSim since the file was last opened (the previous filename is recorded).
- The model file has been archived (the archived filename is recorded).
- One or more of the Simulation Settings have changed.
- A user-defined unit has been created or modified.
- An array label set has been added, deleted or changed.
- A global database download has been carried out.
- A file has been locked onto or unlocked by a Spreadsheet, External or File element.
- One or more options in the Options dialog associated with an extension module have changed.

The following element-specific changes are recorded and stored within the model file when versioning is enabled:

- An element is added, moved, deleted.
- An element is cloned or a cloned element is freed.
- The element’s ID has been changed (the previous ID is recorded).
- The element’s description has changed (the previous description is recorded).
- Any of the element’s attributes or input definitions have changed.
A database download to the element is carried out.
Data is imported into a Lookup Table or a Time Series element.
The checkbox in the Options dialog controlling whether or not Timed Events are mapped to the next timestep has changed.

The following points regarding the changes tracked by GoldSim should be noted:

- A Container is considered to be changed if an element is added to, removed from, or deleted from it.
- Locking/unlocking and sealing a Container are not recorded as changes.
- Purely graphical or cosmetic modifications to the model such as changing an element’s symbol or adding a graphic or text to the graphics pane are not recorded as changes.
- Adding, deleting or modifying Classification categories are not recorded as changes.
- Adding, deleting or editing a Note to an element is not recorded as a change.
- If an element is changed multiple times between versions, GoldSim records that the element has changed, but does not record the number of times or the number of changes that have been made.
- If an element is changed and then changed back (e.g., if the ID is changed and then changed back to its original ID), GoldSim still records this as a change.

In order to display the differences between the current model and any previous version, you must specify the Reference Version in the Version Manager dialog.

Once a Reference Version is specified, GoldSim identifies changed elements in all browsers as follows:

- If an element has been changed since the reference version, the element will appear red in all browsers. (If it is sealed or locked and it has changed, it will appear light red.)
- If any element inside a Container has been modified, but the Container itself has not been modified (e.g., no elements have been added or removed from it), the Container will appear blue in all browsers. (If the Container is sealed or locked and elements within it have been changed, it will appear light blue.)

**Warning:** If a Reference Version is not selected, none of the changed elements will be indicated in the browser. Hence, you must specify a Reference Version in order to view changes.

If you right-click on a colored element (e.g., a red element or a blue Container) in the browser, a menu item, **Show Changes…**, is available toward the top of the menu:
If you select this menu option, the Changes dialog is displayed:

This dialog displays all of the changes that have been made to the element since the specified Reference Version. The Reference Version (selected in the Version Manager) is displayed at the bottom of the dialog. You can change the Reference Version directly from this dialog by selecting a version from the drop-list.

Clicking the Show History checkbox shows a history of all the changes between the Reference Version and the current model.

If the Show History box is cleared, only a summary of the differences between the Reference Version and the current model are shown. Hence, if the element was changed several different times, the Show History display will indicate this,
while the summary display (with Show History cleared) will only indicate that the element has changed since the Reference Version.

Clicking the Show Global Changes checkbox displays the Global changes to the model (in addition to the changes to elements):

If the selected element is a Container, the Changes dialog identifies any elements added to the Container, any elements removed (or deleted) from the Container, and whether any Container properties (e.g., conditionality settings) have been changed.

After an element has been changed, you can add a Change Note to the element that will also be displayed in the Changes dialog.

If an element has been changed since the last (most recent) version stamp, the Version Change Note… option is available in the context menu for the element, and the Change Note… button is available on the Changes dialog for the element. Selecting either of these buttons will display a dialog for adding a change note to the element:
Tracking Model Changes

• Chapter 10: Advanced Modeling Concepts

GoldSim User's Guide

If you add a note, the note is saved with the next version that is created, and is displayed in the element's Changes dialog:

![Version Change Note dialog](image)

Note: A Change Note is intended to be applied for a particular version. Therefore, whenever a new version is created, the element Change Notes for that version are recorded with the version and then immediately cleared. As a result, if you create a subsequent version, the previous note does not appear in the Changes dialog for the new version.

![Changes dialog](image)

Generating a Version Report

In some cases, rather than viewing changes to individual elements, you may want to create a report of all of the changes made between the Reference Version and the current model. You can do so by pressing the Generate Report... button in the Version Manager dialog.

An ASCII text file summarizing all the changes will be generated and immediately opened in the default application associated with .txt files (e.g., Notepad):
This file is written to the directory containing the model file. If it cannot be saved there (due to access issues), GoldSim will save it to the user’s temporary folder (and provide the location of the file in a message).

By default, the name of the Version Report file is “GoldSim Version Difference.txt”.

**Note:** On rare occasions, you may want to instruct GoldSim to insert the model filename into the name of the Version Report filename. To do this, you must edit the Windows Registry. In particular, add a DWORD registry key under HKEY_CURRENT_USER\Software\GTG\Settings named VersionReportEmbedModelName and set it to a non-zero value. If you do so, the name of the Version Report file will be “ModelFilename_VersionReport.txt”. For example, if the model filename was called “Example.gsm”, the Version Report file would be named “Example_VersionReport.txt”.

### Linking Elements to a Database

In simulations which require a great deal of input, it may be desirable (or even mandated) that the simulation model can access the various data sources directly to facilitate and ensure the quality of the data transfer.

One way to accomplish this in GoldSim is to import data from spreadsheets into a Spreadsheet element or into Lookup Tables or Time Series elements.

**Read more:** [Linking a Lookup Table to a Spreadsheet](page 306); [Importing Data into a Time Series from a Spreadsheet](page 194); [Spreadsheet Elements](page 910).

GoldSim also provides a more powerful method. In particular, GoldSim data entry elements can be linked directly to an ODBC-compliant database.
After defining the linkage, you can then instruct GoldSim to download the data at any time. When it does this, GoldSim internally records the time and date at which the download occurred, along with other reference information retrieved from the database (e.g., document references), and this is stored with the model in the Run Log. This information is also displayed in the tool-tip for the linked element.

This allows you to confirm that the correct data were loaded into your model, and provides very strong and defensible quality control over your model input data.

The following elements can be downloaded from a database:

- Data elements;
- 1-D and 2-D Lookup Table elements; and
- Stochastic elements.

There are three steps involved in linking an element to a database:

1. Creating a database which is compatible with GoldSim;
2. Adding the database as a data source to your computer; and
3. Linking an element to the database and downloading the data.

GoldSim can interact with three types of databases: 1) a Generic Database; 2) a Simple GoldSim Database; and 3) an Extended GoldSim Database. Each of these databases have specific formats and different capabilities, as outlined below:

**Generic Database:** Generic databases have a very simple format (consisting of a single table). They can download only to scalar, vector and matrix Data elements. They are the only type of database which can download to vector and matrix Data elements.

**Simple GoldSim Database:** Simple GoldSim databases are more structured than Generic databases. As such they can not only download to scalar, vector and matrix Data elements, but they can also download to Stochastic elements. The database is structured to allow you to store different versions of the same datum (e.g., associated with different dates). You specify from within the database which version is to be used when linked to GoldSim.

**Extended GoldSim Database:** The Extended GoldSim database format is the most complex of the three formats. They can download to scalar, vector and matrix Data elements, Stochastics, and 1-D and 2-D Table elements. The database is structured to allow you to store different versions of the same datum (e.g., associated with different dates). You can specify from within GoldSim which version is to be used when linked to GoldSim.

The table below summarizes the key features of the three database formats:
### Adding Data Sources to Your Computer

<table>
<thead>
<tr>
<th></th>
<th>Download to Scalar Data Elements</th>
<th>Download to Vector and Matrix Data Elements</th>
<th>Download to Stochastics</th>
<th>Download to 1-D and 2-D Tables</th>
<th>Store Different Versions of the Same Datum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple GoldSim</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Extended GoldSim</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

The specific formats for these three types of databases are described in Appendix E.

Before using GoldSim’s database features you must first identify the database(s) you need to access using the Windows Control Panel ODBC Data Sources (32bit) option. This allows you to define a name for each data source, and associate the data source with a specific database file.

For details in adding and configuring data sources, you should refer to your Microsoft Windows documentation. A brief overview is presented below.

To add a data source to your computer, do the following (the instructions here are for Windows 10; they may differ slightly for other Windows versions):

1. From the Windows Control Panel, select under System and Security / Administrative Tools, you will find an option to **Set up ODBC data sources (32-bit)**.

   **Warning:** There will also be an option for 64-bit. It is important that you select the 32-bit option.

2. If your select that option, a dialog similar to the one shown below will be displayed:

   ![ODBC Data Source Administrator](image)

   You can define your database(s) as either ‘User DSN’ or ‘System DSN’, depending whether you want it to be private or public, respectively.
3. Add a data source by pressing the Add… button, which will display a
dialog like this:

![Create New Data Source dialog]

4. Select a driver with the same file extension as your database and press
Finish (the driver selected above is for Microsoft Access). A dialog is
displayed (the dialog will differ for each driver). You will be prompted
to assign a name to the data source, and select the specific file (i.e., a
database file).

Once you have created the data source, you can reference it from within
GoldSim.

### Downloading Element Definitions from a Database

After defining the database and creating the data source, you must then link the
database to specific elements and download the data.

In order to link a specific element to the database, you first select a database
option from the Data Source field in the properties dialog for the element. This
is typically found toward the bottom of the dialog:

![Data Source field in properties dialog]
You can also right-click on the element in the graphics pane or browser, and select **Connect to DataSource...**, which will display a menu for selecting the Data Source.

By default, the Data Source is “None” (i.e., the element is not linked to a database). To link to a database, you must select one of the database types from the drop-down list.

**Note:** Not all three database options are available in the drop-down list for all elements. For example, since a Stochastic cannot be linked to a Generic database, this option is not available for Stochastics in the **Database** tab.

When you do so, a **Database** tab is added to the element.

Three sample GoldSim files (GenericDatabase.gsm, SimpleDatabase.gsm, and Extended_Database.gsm) which illustrates the use of all three database types, are included in the General Examples/Database folder in your GoldSim directory (accessed by selecting **File | Open Example...** from the main menu). This folder also includes three sample databases referenced by these files (GenericDatabase.accdb, SimpleDatabase.accdb, and ExtendedDatabase.accdb). In order to use the GoldSim files, you will need to add the databases as data sources to your computer.

*Read more:* [Adding Data Sources to Your Computer](page 1033).

**Note:** If you open a file which has elements linked to a database which GoldSim cannot find on your machine, it will prompt you to select from a list of databases that are registered on your machine. If none of these databases is appropriate, you must add a new data source to your computer. If you want GoldSim to check to see if it can find all of the databases referenced by elements in your model, press **Model | Database | Validate Databases**.

---

**Downloading from a Generic Database**

The following dialog is displayed when you specify a “Generic Database” for the **Data Source** for an element and click on the **Database** tab in the properties dialog for a Data element (the only element for which this type of data source can be specified):
As described in Appendix E, the Generic database format is very simple, and consists of a single table. To link to the database, you must specifically indicate the record and field you wish to access.

You link and download from a Generic database as follows:

1. Select the appropriate data source from the **Database** drop-down list (all defined data sources will be listed).
2. Select the appropriate table from the database from the **Table** drop-down list.
3. Press the **Define Fields** button in order to access a dialog for specifying which field(s) in the table contain the data. For scalar data, you will need to enter a single field (and this will be indicated at the top of the dialog). Vector and matrix data require multiple fields to be specified.
4. Specify which record (or for matrix data, records) in the table contain the data by specifying an appropriate condition in the **Where** input field.
5. Specify the incoming units of the data in the **Incoming Units** input field (using the appropriate GoldSim abbreviations for the units).
6. Press the **Download Now** button to download the data.

GoldSim notes whether the download was successful (and, if successful, displays the date and time of the download).

If you wish to download a vector, you must specify one record for each item in the vector. You identify the items of the vector using the **Where** input field. You must sort the records in the appropriate order using the **Order By** input field.

If you wish to download a matrix, you must specify one data field for each columns of the matrix (using the Define Fields dialog) and one record for each row of the matrix using the **Where** input field. The order of the fields (i.e., the columns of the matrix) is the order shown in the Define Fields dialog. You can use the **Move Up** button to modify the order. You must sort the records (i.e., the rows of the matrix) in the appropriate order using the **Order By** input field.
The file GenericDatabase.gsm in the General Examples/Database folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) includes examples of how to download a vector and a matrix from a Generic database.

The following dialog is displayed when you specify a “Simple GoldSim Database” for the Data Source for an element and click on the Database tab in the properties dialog for a Data or Stochastic element (the two elements for which this type of data source can be specified):

As described in Appendix E, the Simple GoldSim database has a specific structure in which a string in a field in one of the database tables must uniquely match the ID of the element you wish to link to.

When you link an element to a Simple GoldSim database, GoldSim locates the data to download within the database by using the element's ID and path. Unlike a Generic database, you do not need to manually indicate the record and field you wish to access from within GoldSim.

In addition, the units of data being downloaded are specified directly in the database, and need not be identified on the Database tab (although they must be consistent with the specified dimensions for the element as defined on the Definition tab).

Note: Particular care must be taken when importing Stochastics from a database, as different Stochastics could have different dimensions (e.g., a Binomial is always dimensionless; a Boolean is always a condition). If a Stochastic is defined in your model as a particular type of distribution whose output is inconsistent with that of the distribution that you are trying to import from the database, GoldSim will display an error.
Note: Care must also be taken when importing Stochastics that represent temperatures. This is because temperatures have an absolute reference (i.e., absolute zero). Differences between temperatures are expressed in ‘deg’ units (Cdeg, Fdeg). This can lead to errors when importing Stochastics, since GoldSim assumes a single unit for the distribution parameters, while some types of distributions would require two different types of units. For example, a Normal distribution representing a temperature would require the Mean to be specified in absolute units (e.g., C) and the Standard Deviation to be specified in difference units (e.g., Cdeg). If faced with this problem, there are several approaches for addressing this: 1) Use a distribution type where all parameters have the same units (e.g., triangular, uniform, cumulative); or 2) Specify and import the distributions as dimensionless and then apply a unit to the sampled value using an Expression element.

Read more: Dealing with Temperature Units (page 92).

Because a Simple GoldSim database is so highly structured, if the database is properly defined, linking and downloading from it requires only two steps:

1. Select the appropriate data source from the Database drop-down list (all defined data sources will be listed).
2. Press the Download Now button to download the data.

GoldSim notes whether the download was successful (and, if successful, displays the date and time of the download).

The file SimpleDatabase.gsm in the General Examples/Database folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) includes examples of how to download data from a Simple GoldSim database.

The following dialog is displayed when you specify an “Extended GoldSim Database” for the Data Source for an element and click on the Database tab in the properties dialog for Data, Stochastic or Lookup Table element (the three elements for which this type of data source can be specified):
As described in Appendix E, like the Simple GoldSim database, the Extended GoldSim database has a specific structure in which a string in a field in one of the database tables must uniquely match the ID of the element you wish to link to.

That is, when you link an element to an Extended GoldSim database, GoldSim locates the data to download within the database by using the element's ID. Unlike a Generic database, you do not need to manually indicate the record and field you wish to access from within GoldSim.

In addition, the units of data being downloaded are specified directly in the database, and need not be identified on the Database tab (although they must be consistent with the specified dimensions for the element as defined on the Definition tab).

**Note:** Particular care must be taken when importing Stochastics from a database, as different Stochastics could have different dimensions (e.g., a Binomial is always dimensionless; a Boolean is always a condition). If a Stochastic is defined in your model as a particular type of distribution whose output is inconsistent with that of the distribution that you are trying to import from the database, GoldSim will display an error.

**Note:** Care must also be taken when importing Stochastics that represent temperatures. This is because temperatures have an absolute reference (i.e., absolute zero). Differences between temperatures are expressed in ‘deg’ units (Cdeg, Fdeg). This can lead to errors when importing Stochastics, since GoldSim assumes a single unit for the distribution parameters, while some types of distributions would require two different types of units. For example, a Normal distribution representing a temperature would require the Mean to be specified in absolute units (e.g., C) and the Standard Deviation to be specified in difference units (e.g., Cdeg). If faced with this problem, there are several approaches for addressing this: 1) Use a distribution type where all parameters have the same units (e.g., triangular, uniform, cumulative); or 2) Specify and import the distributions as dimensionless and then apply a unit to the sampled value using an Expression element.

**Read more:** Dealing with Temperature Units (page 92).

Because an Extended GoldSim database is so highly structured, if the database is properly defined, linking and downloading from it requires only two steps:

1. Select the appropriate data source from the Database drop-down list (all defined data sources will be listed).
2. Press the Download Now button to download the data.

GoldSim notes whether the download was successful (and, if successful, displays the date and time of the download).

The More info... button provides some additional information regarding a successful download (the effective date and the Run Log strings associated with the data).

As discussed in detail in Appendix E, for an Extended GoldSim database, you can create multiple records for a particular item of data, each with a different...
Linking Elements to a Database

"Effective Date". This allows you to keep a record of how particular inputs to your model have been modified.

The checkbox allows you to download data associated with a specific Effective Date:

- If checked, you must specify a date, and **Download Now** will download the data with the most recent Effective Date on or before the specified date.
- If cleared, **Download Now** will download the data with the most recent Effective Date on or before the default Effective Date for the element, as specified in the GS_Parameter table of the database (described in Appendix E).
- If cleared and no default Effective Date is specified in the database for the element, **Download Now** will download the data with the most recent Effective Date.

**Note**: As pointed out above, when you link an element to an Extended GoldSim database, GoldSim locates the data to download by using the element’s ID alone (no path is specified in the database). Hence, in order for GoldSim to download the element’s definition from the database, a record in the Element ID field of the database must match the GoldSim element ID exactly (although the comparison is not case-sensitive). This implies that you cannot have multiple elements with identical names (e.g., in two localized containers) which are both being linked to an Extended GoldSim database.

The file `ExtendedDatabase.gsm` in the General Examples/Database folder of your GoldSim directory (accessed by selecting **File | Open Example...** from the main menu) includes examples of how to download data from an Extended GoldSim database.

In addition to downloading each element’s definition individually from a database, you can also globally download all element definitions by pressing the Database Download button from the Standard toolbar:

Yo can also select **Model | Database | Database Download** from the main menu.

If any of your elements are linked to an Extended GoldSim database, you are prompted to specify which Effective Date should be used for each of those elements:

The checkbox activates a Date field:
• If checked, you must specify a date, and each element will download the data with the most recent Effective Date on or before the specified date.

• If cleared, each element will download the data with the most recent Effective Date on or before the default Effective Date for the element, as specified in the GS_Parameter table of the database (described in Appendix E).

• If cleared and no default Effective Date is specified in the database for an element, each element will download the data with the most recent Effective Date.

If one or more of the downloads fail, a dialog will appear summarizing the elements for which the download failed.

While an element is linked to a database, you cannot access or edit the data. After you have downloaded data, however, you can choose to “uncouple” the element from the database by selecting the “No Database” option in the Database tab. When you do this, you will be able to view and edit the values downloaded from the database.

References


Appendix A: Introduction to Probabilistic Simulation

Our knowledge of the way things work, in society or in nature, comes trailing clouds of vagueness. Vast ills have followed a belief in certainty.

Kenneth Arrow, I Know a Hawk from a Handsaw

Appendix Overview

This appendix provides a very brief introduction to probabilistic simulation (the quantification and propagation of uncertainty). Because detailed discussion of this topic is well beyond the scope of this appendix, readers who are unfamiliar with this field are strongly encouraged to consult additional literature. A good introduction to the representation of uncertainty is provided by Finkel (1990) and a more detailed treatment is provided by Morgan and Henrion (1990). The basic elements of probability theory are discussed in Harr (1987) and more detailed discussions can be found in Benjamin and Cornell (1970) and Ang and Tang (1984).

In this Appendix

This appendix discusses the following:

- Types of Uncertainty
- Quantifying Uncertainty
- Propagating Uncertainty
- A Comparison of Probabilistic and Deterministic Analyses
- References
Types of Uncertainty

Many of the features, events and processes which control the behavior of a complex system will not be known or understood with certainty. Although there are a variety of ways to categorize the sources of this uncertainty, for the purpose of this discussion it is convenient to consider the following four types:

- **Value (parameter) uncertainty**: The uncertainty in the value of a particular parameter (e.g., a geotechnical property, or the development cost of a new product);
- **Uncertainty regarding future events**: The uncertainty in the ability to predict future perturbations of the system (e.g., a strike, an accident, or an earthquake);
- **Conceptual model uncertainty**: The uncertainty regarding the detailed understanding and representation of the processes controlling a particular system (e.g., the complex interactions controlling the flow rate in a river); and
- **Numerical model uncertainty**: The uncertainty introduced by approximations in the computational tool used to evaluate the system.

Incorporating these uncertainties into the predictions of system behavior is called *probabilistic analysis* or in some applications, *probabilistic performance assessment*. Probabilistic analysis consists of explicitly representing the uncertainty in the parameters, processes and events controlling the system and propagating this uncertainty through the system such that the uncertainty in the results (i.e., predicted future performance) can be quantified.

Quantifying Uncertainty

When uncertainty is quantified, it is expressed in terms of *probability distributions*. A probability distribution is a mathematical representation of the relative likelihood of an uncertain variable having certain specific values.

There are many types of probability distributions. Common distributions include the normal, uniform and triangular distributions, illustrated below:

![Normal Distribution](image1)
![Uniform Distribution](image2)
![Triangular Distribution](image3)

All distribution types use a set of *arguments* to specify the relative likelihood for each possible value. For example, the normal distribution uses a *mean* and a *standard deviation* as its arguments. The mean defines the value around which the bell curve will be centered, and the standard deviation defines the spread of values around the mean. The arguments for a uniform distribution are a minimum and a maximum value. The arguments for a triangular distribution are a minimum value, a most likely value, and a maximum value.

The nature of an uncertain parameter, and hence the form of the associated probability distribution, can be either *discrete or continuous*. Discrete distributions have a limited (discrete) number of possible values (e.g., 0 or 1; yes
or no; 10, 20, or 30). Continuous distributions have an infinite number of possible values (e.g., the normal, uniform and triangular distributions shown above are continuous). Good overviews of commonly applied probability distributions are provided by Morgan and Henrion (1990) and Stephens et al. (1993).

There are a number of ways in which probability distributions can be graphically displayed. The simplest way is to express the distribution in terms of a **probability density function** (PDF), which is how the three distributions shown above are displayed. In simple terms, this plots the relative likelihood of the various possible values, and is illustrated schematically below:

Note that the “height” of the PDF for any given value is **not** a direct measurement of the probability. Rather, it represents the **probability density**, such that integrating under the PDF between any two points results in the probability of the actual value being between those two points.

Note: Discrete distributions are described mathematically using probability mass functions (pmf), rather than probability density functions. Probability mass functions specify actual probabilities for given values, rather than probability densities.

An alternative manner of representing the same information contained in a PDF is the **cumulative distribution function** (CDF). This is formed by integrating over the PDF (such that the slope of the CDF at any point equals the height of the PDF at that point). For any value on the horizontal axis, the CDF shows the cumulative probability that the variable will be less than or equal to that value. That is, as shown below, a particular point, say [12, 0.84], on the CDF is interpreted as follows: the probability that the value is less than or equal to 12 is equal to 0.84 (84%).
By definition, the total area under the PDF must integrate to 1.0, and the CDF therefore ranges from 0.0 to 1.0.

A third manner of presenting this information is the **complementary cumulative distribution function** (CCDF). The CCDF is illustrated schematically below:

A particular point, say [12, 0.16], on the CCDF is interpreted as follows: the probability that the value is greater than 12 is 0.16 (16%). Note that the CCDF is simply the complement of the CDF; that is, in this example 0.84 is equal to 1 – 0.16.

Probability distributions are often described using **quantiles** or **percentiles** of the CDF. Percentiles of a distribution divide the total frequency of occurrence into hundredths. For example, the 90th percentile is that value of the parameter below which 90% of the distribution lies. The 50th percentile is referred to as the **median**.

**Characterizing Distributions**

Probability distributions can be characterized by their **moments**. The first moment is referred to as the **mean** or **expected value**, and is typically denoted as $\mu$. For a continuous distribution, it is computed as follows:

$$
\mu = \int x f(x) \, dx
$$
where \( f(x) \) is the probability density function (PDF) of the variable. For a
discrete distribution, it is computed as:

\[
\mu = \sum_{i=1}^{N} x_i p(x_i)
\]

in which \( p(x_i) \) is the probability of \( x_i \), and \( N \) is the total number of discrete
values in the distribution.

Additional moments of a distribution can also be computed. The \( n \)th moment of
a continuous distribution is computed as follows:

\[
\mu_n = \int (x - \mu)^n f(x) \, dx
\]

For a discrete distribution, the \( n \)th moment is computed as:

\[
\mu_n = \sum_{i=1}^{N} (x_i - \mu)^n p(x_i)
\]

The second moment is referred to as the variance, and is typically denoted as \( \sigma^2 \). The
square root of the variance, \( \sigma \), is referred to as the standard deviation. The variance and the standard deviation reflect the amount of spread or dispersion in
the distribution. The ratio of the standard deviation to the mean provides a
dimensionless measure of the spread, and is referred to as the coefficient of variance.

The skewness is a dimensionless number computed based on the third moment:

\[
\text{skewness} = \frac{\mu_3}{\sigma^3}
\]

The skewness indicates the symmetry of the distribution. A normal distribution
(which is perfectly symmetric) has a skewness of zero. A positive skewness indicates a shift to the right (and example is the log-normal distribution). A negative skewness indicates a shift to the left.

The kurtosis is a dimensionless number computed based on the fourth moment:

\[
\text{kurtosis} = \frac{\mu_4}{\sigma^4}
\]

The kurtosis is a measure of how "fat" a distribution is, measured relative to a
normal distribution with the same standard deviation. A normal distribution has a kurtosis of zero. A positive kurtosis indicates that the distribution is more "peaky" than a normal distribution. A negative kurtosis indicates that the distribution is "flatter" than a normal distribution.

Given the fact that probability distributions represent the means by which
uncertainty can be quantified, the task of quantifying uncertainty then becomes a
matter of assigning the appropriate distributional forms and arguments to the uncertain aspects of the system. Occasionally, probability distributions can be
defined by fitting distributions to data collected from experiments or other data
collection efforts. For example, if one could determine that the uncertainty in a
particular parameter was due primarily to random measurement errors, one
might simply attempt to fit an appropriate distribution to the available data.

Most frequently, however, such an approach is not possible, and probability
distributions must be based on subjective assessments (Bonano et al., 1989;
Roberds, 1990; Kotra et al., 1996). Subjective assessments are opinions and judgments about probabilities, based on experience and/or knowledge in a
specific area, which are consistent with available information. The process of developing these assessments is sometimes referred to as expert elicitation. Subjectively derived probability distributions can represent the opinions of individuals or of groups. There are a variety of methods for developing subjective probability assessments, ranging from simple informal techniques to complex and time-consuming formal methods. It is beyond the scope of this document to discuss these methods. Roberts (1990), however, provides an overview, and includes a list of references. Morgan and Henrion (1990) also provide a good discussion on the topic.

A key part of all of the various approaches for developing subjective probability assessments is a methodology for developing (and justifying) an appropriate probability distribution for a parameter in a manner that is logically and mathematically consistent with the level of available information. Discussions on the applicability of various distribution types are provided by Harr (1987, Section 2.5), Stephens et al. (1993), and Seiler and Alvarez (1996). Note that methodologies (Bayesian updating) also exist for updating an existing probability distribution when new information becomes available (e.g., Dakins, et al., 1996).

Frequently, parameters describing a system will be correlated (inter-dependent) to some extent. For example, if one were to plot frequency distributions of the height and the weight of the people in an office, there would likely be some degree of positive correlation between the two: taller people would generally also be heavier (although this correlation would not be perfect).

The degree of correlation can be measured using a correlation coefficient, which varies between 1 and -1. A correlation coefficient of 1 or -1 indicates perfect positive or negative correlation, respectively. A positive correlation indicates that the parameters increase or decrease together. A negative correlation indicates that increasing one parameter decreases the other. A correlation coefficient of 0 indicates no correlation (the parameters are apparently independent of each other). Correlation coefficients can be computed based on the actual values of the parameters (which measures linear relationships) or the rank-order of the values of the parameters (which can be used to measure non-linear relationships).

One way to express correlations in a system is to directly specify the correlation coefficients between various model parameters. In practice, however, assessing and quantifying correlations in this manner is difficult. Oftentimes, a more practical way of representing correlations is to explicitly model the cause of the dependency. That is, the analyst adds detail to the model such that the underlying functional relationship causing the correlation is directly represented.

For example, one might be uncertain regarding the solubility of two contaminants in water, while knowing that the solubilities tend to be correlated. If the main source of this uncertainty was actually uncertainty in pH conditions, and the solubility of each contaminant was expressed as a function of pH, the distributions of the two solubilities would then be explicitly correlated. If both solubilities increased or decreased with increasing pH, the correlation would be positive. If one decreased while one increased, the correlation would be negative.

Ignoring correlations, particularly if they are very strong (i.e., the absolute value of the correlation coefficient is close to 1) can lead to physically unrealistic simulations. In the above example, if the solubilities of the two contaminants were positively correlated (e.g., due to a pH dependence), it would be physically inconsistent for one contaminant’s solubility to be selected from the high end of its possible range while the other’s was selected from the low end of its possible range.
When quantifying the uncertainty in a system, there are two fundamental causes of uncertainty which are important to distinguish: 1) that due to inherent variability; and 2) that due to ignorance or lack of knowledge. IAEA (1989) refers to the former as “Type A uncertainty” and the latter as “Type B uncertainty”. These are also sometimes referred to as aleatory and epistemic uncertainty, respectively.

Aleatory uncertainty results from the fact that many parameters are inherently variable (random or noisy) over time such that their behavior can only be described statistically. Examples include the flow rate in a river, the price of a stock or the temperature at a particular location.

Variability in a parameter can be expressed using frequency distributions. A frequency distribution displays the relative frequency of a particular value versus the value. For example, one could sample the flow rate of a river once an hour for a week, and plot a frequency distribution of the hourly flow rate (the x-axis being the flow rate, and the y-axis being the frequency of the observation over the week).

Other parameters are not inherently variable over time, but cannot be specified precisely due to epistemic uncertainty: we lack sufficient information or knowledge to specify their value with certainty. Examples include the strength of a particular material, the mass of a planet, or the efficacy of a new drug.

A fundamental difference between these two types of uncertainty is that epistemic uncertainty (i.e., resulting from lack of knowledge) can theoretically be reduced by studying the parameter or system. That is, since the variability is due to a lack of knowledge, theoretically that knowledge could be improved by carrying out experiments, collecting data or doing research. Aleatory uncertainty, on the other hand, is inherently irreducible. If the parameter itself is inherently variable, studying the parameter further will certainly not do anything to change that variability. This is important because one of the key purposes of probabilistic simulation modeling is not just to make predictions, but to identify those parameters that are contributing the most to the uncertainty in results. If the uncertainty in the results is due primarily to epistemic parameters, we know that we could (at least theoretically) reduce our uncertainty in our results by gaining more information about those parameters.

It should be noted that parameters which have both kinds of uncertainty are not uncommon in simulation models. For example, in considering the flow rate in a river, we know that it will be temporally variable (inherently random in time so it can only be described statistically), but in the absence of adequate data, we will have uncertainty about the statistical measures (e.g., mean, standard deviation) describing that variability. By taking measurements, we can reduce our uncertainty in these statistical measures (i.e., what is the mean flow rate?), but we will not be able to reduce the inherent variability in the flow.

Note that some quantities are variable not over time, but over space or within a collection of items or instances. An example is the age of population. If you had a group of 1000 individuals, you could obtain the age of each individual and create a frequency distribution of the age of the group. This kind of distribution is similar to the example of the flow rate in a river discussed above in that both are described using frequency distributions (one showing a frequency in time, and one showing a frequency of occurrence within a group). The age example, however, is fundamentally different from an inherently random parameter. Whereas a distribution representing an inherently random parameter truly is
describing uncertainty (we cannot predict the value at any given time), a
distribution representing the age distribution is not describing uncertainty at all.
It is simply describing a variability within the group that we could actually
measure and define very precisely.

It is critical not to combine variability like this with uncertainty and represent
both using a single distribution. For example, suppose that you needed to
represent the efficacy of a new drug. The efficacy is different for different age
groups. Moreover, for each age group, there is scientific uncertainty regarding
its efficacy. A common mistake would be to define a single probability
distribution that represents both the variability due to age and the uncertainty
due to lack of knowledge. Not only would it be difficult to define the shape of
such a distribution in the first place, this would produce simulation results that
would be difficult, if not impossible, to interpret in a meaningful way. The
correct way to handle such a situation would be to disaggregate the problem (by
explicitly modeling each age group separately) and then define different
probability distributions for each age group (with each distribution representing
only the scientific uncertainty in the efficacy for that age group).

Propagating Uncertainty

If the inputs describing a system are uncertain, the prediction of the future
performance of the system is necessarily uncertain. That is, the result of any
analysis based on inputs represented by probability distributions is itself a
probability distribution.

In order to compute the probability distribution of predicted performance, it is
necessary to propagate (translate) the input uncertainties into uncertainties in the
results. A variety of methods exist for propagating uncertainty. Morgan and
Henrion (1990) provide a relatively detailed discussion on the various methods.

One common technique for propagating the uncertainty in the various aspects of
a system to the predicted performance (and the one used by GoldSim) is Monte
Carlo simulation. In Monte Carlo simulation, the entire system is simulated a
large number (e.g., 1000) of times. Each simulation is equally likely, and is
referred to as a realization of the system. For each realization, all of the
uncertain parameters are sampled (i.e., a single random value is selected from
the specified distribution describing each parameter). The system is then
simulated through time (given the particular set of input parameters) such that
the performance of the system can be computed.

This results in a large number of separate and independent results, each
representing a possible “future” for the system (i.e., one possible path the system
may follow through time). The results of the independent system realizations are
assembled into probability distributions of possible outcomes. A schematic of
the Monte Carlo method is shown below:
A Comparison of Probabilistic and Deterministic Simulation Approaches

Having described the basics of probabilistic analysis, it is worthwhile to conclude this appendix with a comparison of probabilistic and deterministic approaches to simulation, and a discussion of why GoldSim was designed to specifically facilitate both of these approaches.

The figure below shows a schematic representation of a deterministic modeling approach:

In the deterministic approach, the analyst, although he/she may implicitly recognize the uncertainty in the various input parameters, selects single values...
for each parameter. Typically, these are selected to be “best estimates” or sometimes “worst case estimates”. These inputs are evaluated using a simulation model, which then outputs a single result, which presumably represents a “best estimate” or “worst case estimate”.

The figure below shows a similar schematic representation of a probabilistic modeling approach:

In this case the analyst explicitly represents the input parameters as probability distributions, and propagates the uncertainty through to the result (e.g., using the Monte Carlo method), such that the result itself is also a probability distribution.

One advantage to deterministic analyses is that they can typically incorporate more detailed components than probabilistic analyses due to computational considerations (since complex probabilistic analyses generally require time-consuming simulation of multiple realizations of the system).

Deterministic analyses, however, have a number of disadvantages:

- “Worst case” deterministic simulations can be extremely misleading. Worst case simulations of a system may be grossly conservative and therefore completely unrealistic (i.e., they typically have an extremely low probability of actually representing the future behavior of the system). Moreover, it is not possible in a deterministic simulation to quantify how conservative a “worst case” simulation actually is. Using a highly improbable simulation to guide policy making (e.g., “is the design safe?”) is likely to result in poor decisions.

- “Best estimate” deterministic simulations are often difficult to defend. Because of the inherent uncertainty in most input parameters, defending “best estimate” parameters is often very difficult. In a confrontational environment, “best estimate” analyses will typically evolve into “worst case” analyses.

- Deterministic analyses do not lend themselves directly to detailed uncertainty and sensitivity studies. In order to carry out uncertainty and
sensitivity analysis of deterministic simulations, it is usually necessary to carry out a series of separate simulations in which various parameters are varied. This is time-consuming and typically results only in a limited analysis of sensitivity and uncertainty.

These disadvantages do not exist for probabilistic analyses. Rather than facing the difficulties of defining worst case or best estimate inputs, probabilistic analyses attempt to explicitly represent the full range of possible values. The probabilistic approach embodied within GoldSim acknowledges the fact that for many complex systems, predictions are inherently uncertain and should always be presented as such. Probabilistic analysis provides a means to present this uncertainty in a quantitative manner.

Moreover, the output of probabilistic analyses can be used to directly determine parameter sensitivity. Because the output of probabilistic simulations consists of multiple sets of input parameters and corresponding results, the sensitivity of results to various input parameters can be directly determined. The fact that probabilistic analyses lend themselves directly to evaluation of parameter sensitivity is one of the most powerful aspects of this approach, allowing such tools to be used to aid decision-making.

There are, however, some potential disadvantages to probabilistic analyses that should also be noted:

- **Probabilistic analyses may be perceived as unnecessarily complex, or unrealistic.** Although this sentiment is gradually becoming less prevalent as probabilistic analyses become more common, it cannot be ignored. It is therefore important to develop and present probabilistic analyses in a manner that is straightforward and transparent. In fact, GoldSim was specifically intended to minimize this concern.

- **The process of developing input for a probabilistic analysis can sometimes degenerate into futile debates about the "true" probability distributions.** This concern can typically be addressed by simply repeating the probabilistic analysis using alternative distributions. If the results are similar, then there is not necessity to pursue the "true" distributions further.

- **The public (courts, media, etc.) typically does not fully understand probabilistic analyses and may be suspicious of it.** This may improve as such analyses become more prevalent and the public is educated, but is always likely to be a problem. As a result, complementary deterministic simulations will always be required in order to illustrate the performance of the system under a specific set of conditions (e.g., "expected" or "most likely" conditions).

As this last point illustrates, it is important to understand that use of a probabilistic analysis does not preclude the use of deterministic analysis. In fact, deterministic analyses of various system components are often essential in order to provide input to probabilistic analyses. The key point is that for many systems, deterministic analyses alone can have significant disadvantages and in these cases, they should be complemented by probabilistic analyses.

**References**

The references cited in this appendix are listed below.


Appendix B: Probabilistic Simulation Details

Clever liars give details, but the cleverest don’t.

Anonymous

Appendix Overview

This appendix provides the mathematical details of how GoldSim represents and propagates uncertainty, and the manner in which it constructs and displays probability distributions of computed results. While someone who is not familiar with the mathematics of probabilistic simulation should find this appendix informative and occasionally useful, most users need not be concerned with these details. Hence, this appendix is primarily intended for the serious analyst who is quite familiar with the mathematics of probabilistic simulation and wishes to understand the specific algorithms employed by GoldSim.

In this Appendix

This appendix discusses the following:

- Mathematical Representation of Probability Distributions
- Correlation Algorithms
- Sampling Techniques
- Representing Random (Poisson) Events
- Computing and Displaying Result Distributions
- Computing Sensitivity Analysis Measures
- References
Mathematical Representation of Probability Distributions

The arguments, probability density (or mass) function (pdf or pmf), cumulative distribution function (cdf), and the mean and variance for each of the probability distributions available within GoldSim are presented below.

The beta distribution for a parameter is specified by a minimum value (a), a maximum value (b), and two shape parameters denoted S and T. The beta distribution represents the distribution of the underlying probability of success for a binomial sample, where S represents the observed number of successes in a binomial trial of T total draws.

Alternative formulations of the beta distribution use parameters \( \alpha \) and \( \beta \), or \( \alpha_1 \) and \( \alpha_2 \), where \( S = \alpha = \alpha_1 \) and \( (T-S) = \beta = \alpha_2 \).

Frequently the Beta distribution is also defined in terms of a minimum, maximum, mean, and standard deviation. The shape parameters are then computed from these statistics.

The beta distribution has many variations controlled by the shape parameters. It is always limited to the interval (a,b). Within (a,b), however, a variety of distribution forms are possible (e.g., the distribution can be configured to behave exponentially, positively or negatively skewed, and symmetrically). The distribution form obtained by different S and T values is predictable for a skilled user.

pdf:
\[
f(x) = \frac{1}{B(b-a)^{T-1}}(x-a)^{S-1}(b-x)^{T-S-1}
\]
where:
\[
B = \frac{\Gamma(S)\Gamma(T-S)}{\Gamma(T)}
\]
\[
\Gamma(k) = \int_0^\infty e^{-u}u^{k-1}du
\]
cdf: No closed form
mean:
\[
\mu = a + \frac{S}{T}(b-a)
\]
variance:
\[
\sigma^2 = (b-a)^2 \frac{S(T-S)}{T^2(T+1)}
\]

Note that within GoldSim, there are three ways to define a Beta distribution. You can choose to specify S and T (Beta Distribution). Alternatively, you can specify a mean, standard deviation, minimum and maximum, as defined above (Generalized Beta Distribution). In this case, GoldSim limits the standard deviations that can be specified as follows:

\[
\sigma^* \leq 0.6 \sqrt{\mu^*(1-\mu^*)}
\]
where \( \mu^* = \frac{\mu-a}{b-a}, \quad \sigma^* = \frac{\sigma}{b-a} \)

This constraint ensures that the distribution has a single peak and that it does not have a discrete probability mass at either end of its range.
Finally, you can specify a minimum (a), maximum (b) and most likely value (c) (BetaPERT distribution). In this case, GoldSim assumes shape parameters are as follows:

\[ \alpha = 1 + 4c_n \]
\[ \beta = 5 - 4c_n \]

where

\[ c_n = \frac{c - a}{b - a} \]

Note that in this case, the most likely value specified by the user is not mathematically the most likely value (but is a very close approximation to it).

Note that if the BetaPERT is defined using the 10th and 90th percentile (instead of a minimum and a maximum) the minimum and maximum are estimated through iteration.

### Binomial Distribution

The binomial distribution is a discrete distribution specified by a batch size (n) and a probability of occurrence (p). This distribution can be used to model the number of parts that failed from a given set of parts, where n is the number of parts and p is the probability of the part failing.

pmf:

\[ P(x) = \binom{n}{x} p^x (1 - p)^{n-x} \quad x = 0, 1, 2, 3... \]

where:

\[ \binom{n}{x} = \frac{n!}{x! (n-x)!} \]

cdf:

\[ F(x) = \sum_{i=0}^{x} \binom{n}{i} p^i (1 - p)^{n-i} \]

mean:

\[ np \]

variance:

\[ np(1-p) \]

### Boolean Distribution

The Boolean (or logical) distribution requires a single input: the probability of being true, p. The distribution takes on one of two values: False (0) or True (1).

pmf:

\[ P(x) = \begin{cases} 1-p & x=0 \\ p & x=1 \end{cases} \]

cdf:

\[ F(x) = \begin{cases} 1-p & x=0 \\ 1 & x=1 \end{cases} \]

mean:

\[ \mu = p \]

variance:

\[ \sigma^2 = p(1-p) \]
The cumulative distribution enables the user to input a piece-wise linear cumulative distribution function by simply specifying value \((x_i)\) and cumulative probability \((p_i)\) pairs.

GoldSim allows input of an unlimited number of pairs, \(x_i, p_i\). In order to conform to a cumulative distribution function, it is a requirement that the first probability equal 0 and the last equal 1. The associated values, denoted \(x_0\) and \(x_n\), respectively, define the minimum value and maximum value of the distribution.

**pdf:**
\[
f(x) = \begin{cases} 
0 & x \leq x_0 \text{ or } x \geq x_n \\
\frac{p_{i+1} - p_i}{x_{i+1} - x_i} & x_i \leq x \leq x_{i+1} 
\end{cases}
\]

**cdf:**
\[
F(x) = \begin{cases} 
0 & x \leq x_0 \\
p_i + \left(p_{i+1} - p_i\right)\frac{x - x_i}{x_{i+1} - x_i} & x_i \leq x \leq x_{i+1} \\
1 & x \geq x_n 
\end{cases}
\]

**mean:**
\[
\mu \approx \sum_{i=1}^{n} x_i f(x_i)
\]

**variance:**
\[
\sigma^2 \approx \sum_{i=1}^{n} x_i^2 f(x_i) - \mu^2
\]

The log-cumulative distribution enables the user to input a piece-wise logarithmic cumulative distribution function by simply specifying value \((x_i)\) and cumulative probability \((p_i)\) pairs. Whereas in a cumulative distribution, the density between values is constant (i.e., the distribution between values is uniform), in a log-cumulative, the density of the log of the value is constant (i.e., the distribution between values is log-uniform).

GoldSim allows input of an unlimited number of pairs, \(x_i, p_i\). In order to conform to a cumulative distribution function, it is a requirement that the first probability equal 0 and the last equal 1. The associated values, denoted \(x_0\) and \(x_n\), respectively, define the minimum value and maximum value of the distribution. Also, all values must be positive.
pdf: 
\[ f(x) = \begin{cases} 0 & \text{if } x < x_0 \text{ or } x \geq x_n \\ \frac{P_{i+1} - P_i}{x \ln(x_{i+1} / x_i)} & \text{if } x_i \leq x \leq x_{i+1} \end{cases} \]

cdf: 
\[ F(x) = \begin{cases} 0 & \text{if } x < x_0 \\ p_i \frac{\ln(x_{i+1} / x)}{\ln(x_{i+1} / x_i)} + p_{i+1} \frac{\ln(x / x_i)}{\ln(x_{i+1} / x_i)} & \text{if } x_i \leq x \leq x_{i+1} \\ 1 & \text{if } x \geq x_n \end{cases} \]

mean: 
\[ \mu = \sum_{i=1}^{n} \frac{(P_{i+1} - P_i)(x_{i+1} - x_i)}{\ln(x_{i+1} / x_i)} \]

variance: 
\[ \sigma^2 = \sum_{i=1}^{n} \frac{(P_{i+1} - P_i)(x_{i+1}^2 - x_i^2)}{2 \ln(x_{i+1} / x_i)} \]

**Discrete Distribution**

The discrete distribution enables the user to directly input a probability mass function for a discrete parameter. Each discrete value, \( x_i \), that may be assigned to the parameter, has an associated probability, \( p_i \), indicating its likelihood to occur. To conform to the requirements of a probability mass function, the sum of the probabilities, \( p_i \), must equal 1. The discrete distribution is commonly used for situations with a small number of possible outcomes, such as “flag” variables used to indicate the occurrence of certain conditions.

pmf: 
\[ P(x_i) = p_i \quad \text{if } x = x_i \]

cdf: 
\[ F(x_i) = \sum_{j=i}^{\infty} p_j \]

mean: 
\[ \mu = \sum_{i=1}^{n} x_i p_i \]

variance: 
\[ \sigma^2 = \sum_{i=1}^{n} x_i^2 p_i - \mu^2 \]
Mathematical Representation of Probability Distributions

**Exponential Distribution**

The Exponential distribution is a continuous distribution specified by a mean value (μ) which must be positive. This distribution is typically used to model the time required to complete a task or achieve a milestone.

pdf:
\[ f(x) = \begin{cases} \frac{1}{\mu} e^{-\frac{x}{\mu}} & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \]

cdf:
\[ F(x) = \begin{cases} 1 - e^{-\frac{x}{\mu}} & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \]

mean: \[ \mu \]

variance: \[ \mu^2 \]

**Extreme Probability Distribution**

The Extreme Probability distribution provides the expected extreme probability level of a uniform distribution given a specific number of samples. Both the Minimum and Maximum Extreme Probability Distributions are equivalent to the Beta distribution with the following parameters:

<table>
<thead>
<tr>
<th></th>
<th>Maximum</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>Number of samples</td>
<td>1</td>
</tr>
<tr>
<td>(T-S)</td>
<td>1</td>
<td>Number of Samples</td>
</tr>
</tbody>
</table>

pdf:
\[ f(x) = \frac{1}{B(b-a)^{T-1}} (x-a)^{S-1} (b-x)^{T-S-1} \]

where:
\[ B = \frac{\Gamma(S)\Gamma(T-S)}{\Gamma(T)} \]

\[ \Gamma(k) = \int_0^\infty u^{k-1} du \]

cdf: No closed form

mean:
\[ \mu = a + \frac{S}{T}(b-a) \]

variance:
\[ \sigma^2 = (b-a)^2 \frac{S(T-S)}{T^2(T+1)} \]
### Extreme Value Distribution

The Extreme Value distribution (also known as the Gumbel distribution) is used to represent the maximum or minimum expected value of a variable. It is specified with the mode (m) and a scale parameter (s) that must be positive.

<table>
<thead>
<tr>
<th></th>
<th>Maximum</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>pdf</td>
<td>( f(x) = \frac{1}{s} e^{-\frac{z}{s}} )</td>
<td>( f(x) = \frac{1}{s} e^{-\frac{(x-m)}{s}} )</td>
</tr>
<tr>
<td>cdf</td>
<td>( F(x) = e^{-\frac{z}{s}} )</td>
<td>( F(x) = 1 - e^{-\frac{(x-m)}{s}} )</td>
</tr>
<tr>
<td>mean</td>
<td>( \mu = m + 0.57722 \sigma )</td>
<td>( \mu = m - 0.57722 \sigma )</td>
</tr>
<tr>
<td>variance</td>
<td>( \sigma^2 = \frac{6}{\pi^2} )</td>
<td>( \sigma^2 = \frac{6}{\pi^2} )</td>
</tr>
</tbody>
</table>

### Gamma Distribution

The gamma distribution is most commonly used to model the time to the \( k \)th event, when such an event is modeled by a Poisson process with rate parameter \( \lambda \). Whereas the Poisson distribution is typically used to model the number of events in a period of given length, the gamma distribution models the time to the \( k \)th event (or alternatively the time separating the \( k \)th and \( k+1 \) events).

The gamma distribution is specified by the Poisson rate variable, \( \lambda \), and the event number, \( k \). The random variable, denoted as \( x \), is the time period to the \( k \)th event. Within GoldSim, the gamma distribution is specified by the mean and the standard deviation, which can be computed as a function of \( \lambda \) and \( k \).

\[
\text{pdf: } f(x) = \frac{\lambda^k x^{k-1} e^{-\lambda x}}{\Gamma(k)}
\]

\[
\text{cdf: } F(x) = \frac{\Gamma(k, \lambda x)}{\Gamma(k)}
\]

where:
\[
\Gamma(k) = \int_0^\infty e^{-u} u^{k-1} du \quad \text{(gamma function)}
\]

\[
\Gamma(k, x) = \int_x^\infty e^{-u} u^{k-1} du \quad \text{(incomplete gamma function)}
\]

\[
k = \frac{\mu^2}{\sigma^2}
\]

\[
\lambda = \frac{\mu}{\sigma^2}
\]
Mathematical Representation of Probability Distributions

mean: \( \mu \)

variance: \( \sigma^2 \)

k and \( \lambda \) are also sometimes called \( \alpha \) and \( \beta \), respectively, and referred to as the shape and scale parameters.

Note that

mean: \( \frac{\alpha}{\beta} \)

variance: \( \frac{\alpha}{\beta^2} \)

If \( \alpha \) is near zero, the distribution is highly skewed. For \( \alpha=1 \), the gamma distribution reduces to an exponential(\( \beta^{-1} \)) distribution. If \( \alpha=n/2 \) and \( \beta = 1/2 \), the distribution is known as a chi-squared distribution with \( n \) degrees of freedom.

The log-normal distribution is used when the logarithm of the random variable is described by a normal distribution. The log-normal distribution is often used to describe environmental variables that must be positive and are positively skewed.

In GoldSim, the log-normal distribution may be based on either the true mean and standard deviation, or on the geometric mean (identical to the median) and the geometric standard deviation. Thus, if the variable \( x \) is distributed log-normally, the mean and standard deviation of \( \log x \) may be used to characterize the log-normal distribution. (Note that either base 10 or base \( e \) logarithms may be used).

pdf:

\[
f(x) = \frac{1}{\zeta x \sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{\ln(x) - \lambda}{\zeta} \right)^2}
\]

where:

\[
\zeta^2 = \ln \left[ 1 + \left( \frac{\sigma}{\mu} \right)^2 \right] \quad \text{(variance of } \ln x \text{)}
\]

\( \zeta \) is referred to as the shape factor; and

\[
\lambda = \ln(\mu) - \frac{1}{2} \zeta^2 \quad \text{(expected value of } \ln x \text{)}
\]

cdf: No closed form solution

mean (arithmetic):

\[
\mu = \exp \left[ \lambda + \frac{1}{2} \zeta^2 \right]
\]

The mean computed by the above formula is the expected value of the log-normally distributed variable \( x \) and is a function of the mean and standard deviation of \( \ln x \). The mean value can be estimated by the arithmetic mean of a sample data set.

variance (arithmetic):

\[
\sigma^2 = \mu^2 \left[ \exp \left( \zeta^2 \right) - 1 \right]
\]

The variance computed by the above formula is the variance of the log-normally distributed variable \( x \). It is a function of the mean of \( x \) and the standard deviation of \( \ln x \). The variance of \( x \) can be estimated by the sample variance computed arithmetically.

Other useful formulas:
Geometric mean = $e^\lambda$
Geometric standard deviation = $e^{\zeta}$

A commonly used descriptor for a log-normal distribution is its Error Factor (EF), where the EF is defined as (geometric standard deviation) ^ 1.645. 90% of the distribution lies between Median/EF and Median*EF.

The negative binomial distribution is a discrete distribution specified by a number of successes (n), which can be fractional, and a probability of success (p). This distribution can be used to model the number of failures that occur when trying to achieve a given number of successes, and is used frequently in actuarial models.

**pmf:**

$$P(x) = \binom{x + n - 1}{x} p^n (1 - p)^x$$

where:

$$\binom{x + n - 1}{x} = \frac{(x + n - 1)!}{x!(n - 1)!}$$

**cdf:**

$$F(x) = p^n \sum_{i=0}^{x} \binom{i + n - 1}{i} (1 - p)^i$$

**mean:**

$$\frac{n(1 - p)}{p}$$

**variance:**

$$\frac{n(1 - p)}{p^2}$$

---

**Normal Distribution**

The normal distribution is specified by a mean ($\mu$) and a standard deviation ($\sigma$). The linear normal distribution is a bell shaped curve centered about the mean value with a half-width of about four standard deviations. Error or uncertainty that can be higher or lower than the mean with equal probability may be satisfactorily represented with a normal distribution. The uncertainty of average values, such as a mean value, is often well represented by a normal distribution, and this relation is further supported by the Central Limit Theorem for large sample sizes.

**pdf:**

$$f(x) = \frac{1}{\sqrt{2\pi \sigma^2}} e^{-\frac{1}{2} \left( \frac{x - \mu}{\sigma} \right)^2}$$

**cdf:**

No closed form solution

**mean:**

$$\mu$$

**variance:** $\sigma^2$
**Pareto Distribution**

The Pareto distribution is a continuous, long-tailed distribution specified by a shape parameter (a) and a scale parameter (b). The shape parameter and the scale parameter must be greater than zero. This distribution can be used to model things like network traffic in a telecommunications system or income levels in a particular country.

pdf: \[ f(x) = \begin{cases} \frac{ab^a}{x^{a+1}} & \text{if } x \geq b \\ 0 & \text{otherwise} \end{cases} \]

cdf: \[ F(x) = \begin{cases} 1 - \left(\frac{b}{x}\right)^a & \text{if } x \geq b \\ 0 & \text{otherwise} \end{cases} \]

mean: \[ \frac{ab}{a-1} \]

variance: \[ \frac{ab^2}{(a-1)^2(a-2)} \]

**Pearson Type III Distribution**

Often used in financial and environmental modeling, the Pearson Type III distribution is a continuous distribution specified by location (α), scale (β) and shape (p) parameters. Both the scale and shape parameters must be positive.

Note that the Pearson Type III distribution is equivalent to a gamma distribution if the location parameter is set to zero.

pdf: \[ f(x) = \begin{cases} \frac{1}{\beta \Gamma(p)} \left(\frac{x-\alpha}{\beta}\right)^{p-1} e^{-\left(\frac{x-\alpha}{\beta}\right)} & \text{if } x \geq \alpha \\ 0 & \text{if } x < \alpha \end{cases} \]

cdf: \[ F(x) = \frac{\Gamma\left(p, \frac{x-\alpha}{\beta}\right)}{\Gamma(p)} \]

mean: \[ \mu = \alpha + p\beta \]

variance: \[ \sigma^2 = p\beta^2 \]
Poisson Distribution

The Poisson distribution is a discrete distribution specified by a mean value, \( \mu \). The Poisson distribution is most often used to determine the probability for one or more events occurring in a given period of time. In this type of application, the mean is equal to the product of a rate parameter, \( \lambda \), and a period of time, \( \omega \). For example, the Poisson distribution could be used to estimate probabilities for numbers of earthquakes occurring in a 100 year period. A rate parameter characterizing the number of earthquakes per year would be needed for input to the distribution. The time period would simply be equal to 100 years.

pdf:
\[
f(x) = \frac{e^{-\mu} \mu^x}{x!}
\]

cdf:
\[
F(x) = e^{-\mu} \sum_{i=0}^{x} \frac{\mu^i}{i!}
\]

mean:
\[
\mu = \lambda \omega
\]

variance:
\[
\sigma^2 = \mu
\]

where \( \lambda \) and \( \omega \) are the “rate” and “time period” parameters, respectively. Note that quotations are used because the terminology rate and time period applies to only one application of the Poisson distribution.

Student’s t Distribution

The Student’s t distribution requires a single input: the number of degrees of freedom, which equals the number of samples minus one.

mean:
\[
0
\]

variance:
\[
\frac{v}{v - 2}
\]

where \( v \) is the number of degrees of freedom.

Sampled Result Distribution

The sampled result distribution allows you to construct a distribution using observed results. GoldSim generates a CDF by sorting the observations and assuming that a cumulative probability of \( 1/(\text{Number of Observations}) \) exists between each data point. If there are multiple data points at the same value, a discrete probability equal to \( (N)/(\text{Number of Observations}) \) is applied at the value, where \( N \) is equal to the number of identical observations.

If the Extrapolation option is cleared, a discrete probability of \( 0.5/(\text{Number of observations}) \) is assigned to the minimum and maximum values. When the extrapolation option is selected, GoldSim extends the generated CDF to cumulative probability levels of 0 and 1 using the slope between the two smallest and two largest unique observations.
The triangular distribution is specified by a minimum value (a), a most likely value (b), and a maximum value (c).

pdf: \[ f(x) = \begin{cases} \frac{2(x-a)}{(b-a)(c-a)} & a \leq x \leq b \\ \frac{2(c-x)}{(c-b)(c-a)} & b \leq x \leq c \\ 0 & x < a \text{ or } x > c \end{cases} \]

cdf: \[ F(x) = \begin{cases} 0 & x < a \\ \frac{(x-a)^2}{(b-a)(c-a)} & a \leq x \leq b \\ 1 - \frac{(c-x)^2}{(c-b)(c-a)} & b < x < c \\ 1 & x \geq c \end{cases} \]

mean: \[ \mu = \frac{a + b + c}{3} \]

variance: \[ \sigma^2 = \frac{a^2 + b^2 + c^2 - ab - ac - bc}{18} \]

Note that if the triangular is defined using the 10th and 90th percentile (instead of a minimum and a maximum) the minimum and maximum are estimated through iteration.
Log-Triangular Distribution

The log-triangular distribution is used when the logarithm of the random variable is described by a triangular distribution. The minimum (a), most likely (b), and maximum (c) values are specified in linear space.

pdf: \[ f(x) = \begin{cases} \frac{2}{\ln(b/a) \ln(c/a)} \left( \frac{1}{x} \right) & \text{if } a \leq x \leq b \\ \frac{2}{\ln(c/a) \ln(c/b)} \left( \frac{1}{x} \right) & \text{if } b \leq x \leq c \\ 0 & \text{otherwise} \end{cases} \]

cdf: \[ F(x) = \begin{cases} 0 & \text{if } x < a \\ \frac{(\ln(x/a))^2}{\ln(b/a) \ln(c/a)} & \text{if } a \leq x \leq b \\ 1 - \frac{(\ln(c/x))^2}{\ln(c/a) \ln(c/b)} & \text{if } b < x \leq c \\ 1 & \text{if } x > c \end{cases} \]

mean: \[ \mu = \frac{2}{d_1} \left( a + b \left( \ln \left( \frac{b}{a} \right) - 1 \right) \right) + \frac{2}{d_2} \left( c + b \left( \ln \left( \frac{b}{c} \right) - 1 \right) \right) \]

variance: \[ \sigma^2 = \frac{2}{d_1} \left( \frac{a^2}{4} + \frac{b^2}{2} \left( \ln \left( \frac{b}{a} \right) - \frac{1}{2} \right) \right) + \frac{2}{d_2} \left( \frac{c^2}{4} + \frac{b^2}{2} \left( \ln \left( \frac{b}{c} \right) - \frac{1}{2} \right) \right) - \mu^2 \]

where: \[ d_1 = \ln \left( \frac{c}{a} \right) \ln \left( \frac{b}{a} \right) \]
\[ d_2 = \ln \left( \frac{c}{a} \right) \ln \left( \frac{c}{b} \right) \]

Note that if the log-triangular is defined using the 10th and 90th percentile (instead of a minimum and a maximum) the minimum and maximum are estimated through iteration.
The uniform distribution is specified by a minimum value (a) and a maximum value (b). Each interval between the endpoints has equal probability of occurrence. This distribution is used when a quantity varies uniformly between two values, or when only the endpoints of a quantity are known.

pdf: \[ f(x) = \frac{1}{b-a} \quad a \leq x \leq b \]
\[ 0 \quad \text{otherwise} \]
cdf: \[ F(x) = \begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & a \leq x \leq b \\ 1 & x > b \end{cases} \]
mean: \[ \mu = \frac{b+a}{2} \]
variance: \[ \sigma^2 = \frac{(b-a)^2}{12} \]

The log-uniform distribution is used when the logarithm of the random variable is described by a uniform distribution. Log-uniform is the distribution of choice for many environmental parameters that may range in value over two or more log-cycles and for which only a minimum value and a maximum value can be reasonably estimated. The log-uniform distribution has the effect of assigning equal probability to the occurrence of intervals within each of the log-cycles. In contrast, if a linear uniform distribution were used, only the intervals in the upper log-cycle would be represented uniformly.

pdf: \[ f(x) = \frac{1}{x(\ln b - \ln a)} \quad a \leq x \leq b \]
\[ 0 \quad x \leq a \text{ or } x \geq b \]
cdf: \[ F(x) = \begin{cases} 0 & x \leq a \\ \frac{\ln x - \ln a}{\ln b - \ln a} & a \leq x \leq b \\ 1 & x > b \end{cases} \]
mean: \[ \mu = \frac{b-a}{\ln b - \ln a} \]
variance: \[ \sigma^2 = \frac{b^2 - a^2}{2(\ln b - \ln a)} - \left( \frac{b-a}{\ln b - \ln a} \right)^2 \]
The Weibull distribution is typically specified by a minimum value ($\varepsilon$), a scale parameter ($\beta$), and a slope or shape parameter ($\alpha$). The random variable must be greater than 0 and also greater than the minimum value, $\varepsilon$.

The Weibull distribution is often used to characterize failure times in reliability models. However, it can be used to model many other environmental parameters that must be positive. There are a variety of distribution forms that can be developed using different values of the distribution parameters.

**pdf:**
$$f(x) = \frac{\alpha}{\beta - \varepsilon} \left( \frac{x - \varepsilon}{\beta - \varepsilon} \right)^{\alpha-1} e^{-\left( \frac{x - \varepsilon}{\beta - \varepsilon} \right)^\alpha}$$

**cdf:**
$$F(x) = 1 - e^{-\left( \frac{x - \varepsilon}{\beta - \varepsilon} \right)^\alpha}$$

**mean:**
$$\mu = \varepsilon + (\beta - \varepsilon) \Gamma\left(1 + \frac{1}{\alpha}\right)$$

**variance:**
$$\sigma^2 = (\beta - \varepsilon)^2 \left[ \Gamma\left(1 + \frac{2}{\alpha}\right) - \Gamma^2\left(1 + \frac{1}{\alpha}\right) \right]$$

The Weibull distribution is sometimes specified using a shape parameter, which is simply $\beta - \varepsilon$. Within GoldSim, the Weibull is defined by $\varepsilon$, $\alpha$, and the mean-$\varepsilon$. As shown above, the mean can be readily computed as a function of $\varepsilon$, $\alpha$, and $\beta$.

In practice, the Weibull distribution parameters are moderately difficult to determine from sample data. The easiest approach utilizes the cdf, fitting a regression through the sample data to estimate $\alpha$ (regression slope) and the difference quantity, $\beta - \varepsilon$.

Several distributions in GoldSim can be truncated at the ends (normal, log-normal, Gamma, and Weibull). That is, by specifying a lower bound and/or an upper bound, you can restrict the sampled values to lie within a portion of the full distribution’s range.

The manner in which truncated distributions are sampled is straightforward. Because each point in a full distribution corresponds to a specific cumulative probability level between 0 and 1, it is possible to identify the cumulative probability levels of the truncation points. These then define a scaling function which allows sampled values to be mapped into the truncated range.

In particular, suppose the cumulative probability levels for the lower bound and upper bound were $L$ and $U$, respectively. Any sampled random number $R$ (representing a cumulative probability level between 0 and 1) would then be scaled as follows:

$$L + R(U-L)$$

This resulting "scaled" cumulative probability level would then be used to compute the sampled value for the distribution. The scaling operation ensures that it falls within the truncated range.
Correlation Algorithms

Several GoldSim elements that are used to represent uncertainty or stochastic behavior in models permit the user to define correlations between elements or amongst the items of a vector-type element.

To generate sampled values that reflect the specified correlations GoldSim uses copulas and the Iman and Conover methodology.

A copula is a function that joins two or more univariate distributions to form a multivariate distribution. As such, it provides a method for specifying the correlation between two variables. Copulas are described in general terms by Dorey (2006) and in detail by Embrechts, Lindskog and McNeil (2001).

GoldSim uses two different copulas to generate correlated values: the Gaussian copula and the t-distribution copula. When a Stochastic element is correlated to itself, or to another Stochastic element, GoldSim uses the Gaussian copula to generate the correlated value. A vector-type Stochastic or History Generator can use either the Gaussian copula or a t-distribution copula to generate correlated values.

The Gaussian copula produces values where the correlation between variables is stronger towards the middle of the distributions than it is at the tails. The plot below shows the values for two variables (uniform distributions between 0 and 1) generated using the Gaussian copula with a correlation coefficient of 0.9:

A t-distribution copula produces a correlation that is stronger at the tails than in the middle. The plot below shows the values for the two variables generated using the t-distribution copula for the same variables with a correlation coefficient of 0.9 and the Degrees of Freedom setting in the copula of 1):
The t-distribution’s form is often what is observed in the real world: correlations at the extremes (e.g. representing a rare, but significant event such as a war) tend to be higher than correlations in the middle (representing the higher variability in more common occurrences).

The Degrees of Freedom setting controls the tail dependency in the copula. A low value produces stronger dependence in the tails, while higher values produce stronger correlations in the middle of the distributions. This means that a t-distribution copula with a high number of Degrees of Freedom will begin to behave like a Gaussian copula.

One of the weaknesses of the copula approach to generating correlated samples is that it does not respect Latin Hypercube Sampling (with the exception of the first item in a vector-type stochastic where the Gaussian copula is used to generate sampled values).

The Iman and Conover approach is designed to produce a set of correlated items that each respect Latin Hypercube sampling. Complete details on the algorithm’s methodology can be found in Iman and Conover (1982).

Its behavior is similar, but not identical, to a Gaussian for the first sample:

However, if the element is resampled during a realization, elements that use the Iman and Conover approach will use the Gaussian copula to generate the second and subsequent sets of sampled data.
Sampling Techniques

This section discusses the techniques used by GoldSim to sample elements with random behavior. These include the following GoldSim elements:

- Stochastic
- Random Choice
- Timed Event Generator
- Event Delay
- Discrete Change Delay
- History Generator
- Source (in the Radionuclide Transport Module)
- Action element (in the Reliability module)
- Function element (in the Reliability module)

After first discussing how GoldSim generates random numbers in order to sample these elements, two enhanced sampling techniques provided by GoldSim (Latin Hypercube sampling and importance sampling) are discussed.

Generating Random Numbers to Sample Elements

In order to sample an element (we will simplify the discussion here by using a Stochastic element as an example rather than one of the other types of elements), GoldSim starts with the CDF of the distribution that we want to sample. Below is a CDF for a Normal distribution with a mean of 10m and a standard deviation of 2m:

To randomly sample this distribution, we simply need to do the following:

1. Obtain a random number (i.e., a number between 0 and 1).
2. Use the CDF to map that random number to the corresponding sampled value.

So, for example, in the CDF above, a random number of about 0.2 would correspond to a sampled value of about 8.3m.

As can be seen, this sampling process itself is conceptually very simple. The more complicated part involves obtaining the random number needed to sample the element. That is, in order to carry out Monte Carlo simulation, GoldSim (and
any Monte Carlo simulator) needs to consistently generate a series of random numbers.

In GoldSim, the process consists of the following components:

- Several different types of random number seeds. You can simply think of a random number seed as an integer number. It actually consists of a pair of long (32-bit) integers, but that is not important to the discussion that follows.

- Random numbers. A random number, as used here, has a specific definition: it is a real number between 0 and 1.

- A seed generator. This is an algorithm that takes as input one random number seed and randomly generates a new random number seed. A particular value for the input seed always generates the same output seed, but different input seeds generate different output seeds.

- A random number generator. This is another algorithm. It takes as input one random number seed and generates a random number. A particular value for the random number seed always generates the same random number, but different random number seeds generate different random numbers.

Within GoldSim, there are several types of random number seeds:

- The model itself (as well as any SubModel) has a run seed. If you choose to Repeat Sampling Sequences (an option on the Monte Carlo tab of the Simulation Settings dialog), this seed is (reproducibly) created based on an integer number that can be edited by the user. If you do not Repeat Sampling Sequences, it is randomly created based on the computer’s system clock.

- The realization seed is a mutable seed that is initialized with the run seed and then updated for each realization.

- Each Stochastic element (as well as other elements that behave probabilistically) has its own random number seed. This is referred to as the element seed. This seed is created (in a random manner, based on the system clock) when the element is first created.

Note: No two elements in a GoldSim model can have the same element seed. This means that if an element is copied and pasted into a model where no elements have the same seed value, its seed will be unchanged. However, if it is pasted into the same model, or into a model where another element already has that seed value, one of the elements with the same seed value will be given a new unique seed. (Element seeds can be displayed by selecting the element in the graphics pane and pressing Ctrl-Alt-Shift-F12).

- For every element that behaves probabilistically), the realization seed and the element seed are combined together to create a combined seed for that element. It is this combined seed that is used to generate a random number using a random number generator.

This random number seed structure is illustrated below:
The *run seed* and *element seeds* are constant during a simulation (they never change). The run seed is marked using a dashed line to indicate that although it is constant during a simulation, it can be changed by the user. The element seed (which is different for each element $j$) is created when the element is created and cannot be changed. As we shall see below, however, the *realization seeds* and *combined seeds* are not constant during a simulation but change as the simulation proceeds.

So given all of this information, let’s describe how GoldSim carries out a Monte Carlo simulation by considering a very simple model consisting of a single Stochastic element that is resampled every day, with the model being run for multiple realizations:

1. At the beginning of the simulation, GoldSim has a value for the *run seed*, and a single *element seed*.
2. At the beginning of the simulation (assuming we are repeating sampling sequences), the *run seed* is used to initialize a *realization seed*.
3. At the beginning of each realization, we do the following:
   a. The current *realization seed* is input into the seed generator to generate a new *realization seed* for this realization.
   b. The *realization seed* is combined with the *element seed* to create a *combined seed* for the element.
   c. Now that we have the *combined seed*, two things happen:
      i. The *combined seed* is input into the random number generator to generate a *random number* for the element. This random number is then mapped to the CDF to obtain a *sampled value* for the element.
      ii. The *combined seed* is input into the seed generator to generate a new *combined seed* that we will use the next time we need a random number.
   d. Whenever we need to resample the element during the realization (in this case every day), we need a new random number. To do so, every day we repeat steps i and ii above.

This same logic is shown schematically below:
GoldSim’s random number generation process is based on a method presented by L’Ecuyer (1988). As pointed out above, each seed actually consists of two long (32-bit) integers. When random numbers are generated, each of the integers cycles through a standard linear congruential sequence, with different constants used for the two sequences (so that their repeat-cycles are different). The random numbers that are generated are a function of the combination of the two seeds, so that the length of their repeat period is extremely long.

Now let’s consider the various options on the Monte Carlo tab of the Simulation Settings dialog:
In particular, we will focus on just two fields: Repeat Sampling Sequences and Random Seed. These two fields impact the run seed in the following ways:

- If Repeat Sampling Sequences in checked, you can specify a Random Seed. The Random Seed is used to create the run seed.

- If Repeat Sampling Sequences is cleared, the run seed is created “on the fly” using the system clock. As a result, it is different every time the model is run.

These facts, combined with the logic outlined above, can be used to describe exactly how elements will be sampled in various models under any set of circumstances. In particular:

- If Repeat Sampling Sequences is checked (the default), as long as you do not modify the model, you will get the same results (i.e., the same random numbers will be used) if you run the model today, and then run it again tomorrow. This is because the run seed is unchanged.

- Similarly, if Repeat Sampling Sequences is checked and you copy a model to someone else (and they do not make any changes), they will get the same results as you.

- If Repeat Sampling Sequences is checked, but the Random Seed is changed (e.g., from 1 to 2), you will get different results (i.e., different random numbers will be used). This is because the run seed is different. If you then change the Random Seed back to the original value, you will reproduce the original results.

- If Repeat Sampling Sequences is cleared, every time you run the model you will get different results (i.e., different random numbers will be used). This is because the run seed is different.
• If two people simultaneously build the same simple model with the exact same inputs (including the same Random Seed), the results will still be different (i.e., different random numbers will be used). This is because the element seeds will be different.

• If the user selects the option to Run the following Realization only, such as realization 16, GoldSim simply iterates through the process the necessary number of times prior to starting the specified realization.

GoldSim provides an option to implement a Latin Hypercube sampling (LHS) scheme (in fact, it is the default when a new GoldSim file is created). The LHS option results in forced sampling from each “stratum” of each parameter.

The following elements use LHS sampling:

• Stochastic
• Random Choice
• Timed Event Generator
• Time Series (when time shifting using a random starting point)
• Action (in the Reliability Module)
• Function (in the Reliability Module)

Each element’s probability distribution (0 to 1) is divided into up to 10000 equally likely strata or slices (actually, the lesser of the number of realizations and 10000). The strata are then “shuffled” into a random sequence, and a random value is then picked from each stratum in turn. This approach ensures that a uniform spanning sampling is achieved.

Note: If possible, GoldSim will attempt to create LHS sequences where subsets are also complete LHS sequences. This means that if the total number of realizations is an even number, the first half and second half of the realizations are complete LHS sequences. If the total number of realizations is divisible by 4 or 8, that fraction of the total number of realizations, run in sequence, will be complete LHS sequences. This property of GoldSim’s LHS sequences is sometimes useful for statistical purposes and also permits a user to extract a valid LHS sequence from a partially completed simulation by screening realizations. The details of this approach are discussed below (“Latin Hypercube Subsets”).

Note that each element has an independent sequence of shuffled strata that are a function of the element’s internal random number seed and the number of realizations in the simulation. If the number of realizations exceeds 10,000, then at the 10,001st realization each element makes a random jump to a new position in its strata sequence. A random jump is repeated every 10,000 realizations.

If you select “Use mid-points of strata” in the Simulation Settings dialog in models with less than 10,000 realizations, GoldSim will use the expected value of the strata selected for that realization. (Even if this option is selected, in simulations with greater than 10,000 realizations, the 10,001 and subsequent realizations will use random values from within the strata selected for that realization.) Using mid-points provides a slightly better representation of the distribution, but without the full randomness of the original definition of Latin Hypercube sampling (as described by McKay, Conover and Beckman, 1979).
If you select “Use random points in strata” in the Simulation Settings dialog, GoldSim also picks a random value from each stratum.

**Note:** LHS is only applied to the first sampled random value in each realization. Subsequent samples will not use LHS.

LHS appears to have a significant benefit only for problems involving a few independent stochastic parameters, and with moderate numbers of realizations. In no case does it perform worse than true random sampling, and accordingly LHS sampling is the default for GoldSim.

Note that the binary subdivision approach (described in more detail below) and the use of mid-stratum values are GoldSim-specific modifications to the original description of Latin Hypercube Sampling, as described in McKay, Conover and Beckman (1979).

In order to allow users to do convergence tests, GoldSim’s LHS sampling automatically organizes the LHS strata for each random variable so that binary subsets of the overall number of realizations each represent an independent LHS sample over the full range of probabilities.

For example, if the user does 1,000 realizations, GoldSim will generate strata such that:

- Realizations 1-125 represent a full LHS sample with 125 strata.
  Realizations 126-250, 251-375, etc. through 876-1000 also represent full LHS samples with 125 strata each.
- Also, realizations 1-250, 251-500, 501-750, and 751-1000 represent full LHS samples with 250 strata each.
- And, realizations 1-500 and 501-1000 represent full LHS samples with 500 strata each.

The generation of binary subsets is automatic, and is carried out whenever the total number of realizations is an even number. Up to 16 binary subsets will be generated, if the number of realizations can be subdivided evenly four times.

For example, if the total number of realizations was 100 then GoldSim would generate 2 subsets of 50 strata each and 4 subsets of 25 strata. If the total number of realizations was 400 then GoldSim would generate 2 subsets of 200 strata, 4 subsets of 100 strata, 8 subsets of 50 strata, and 16 subsets of 25 strata.

The primary purpose of this sampling approach is to use the subsets to carry out statistical tests of convergence. For example, the mean of each of the subsets of results could be evaluated and a t-test used to estimate statistics of the population mean, as described in Iman (1982). Rather than carrying out a set of independent LHS simulations using different random seeds, this approach allows the user to run a single larger simulation, with the benefits of a better overall representation of the system’s result distribution, while still being able to test for convergence and to generate confidence bounds on the results. (A secondary benefit of the binary sampling approach is that if a simulation is terminated partway through it should have a slightly greater likelihood of having uniform sampling over the completed realizations than normal Latin Hypercube sampling would.)

The algorithm for assigning strata to the binary subsets is quite simple. For each pair of strata (e.g., 1st and 2nd, 3rd and 4th), the first member of the pair is randomly assigned to one of two “piles” (“left” or “right”), and the second member is assigned to the opposite pile. That is, conceptually, it can be
imagined that the full set of strata is sent one at a time to a ‘flipper’ that randomly chooses ‘left’ or ‘right’ on its first, third, fifth etc. activations, and on its second, fourth, sixth etc. activations chooses the opposite of the previous value.

For the case of one binary subdivision of a total of N\textsubscript{s} strata, the algorithm goes through the strata sequentially from lowest to highest, and passes them to a flipper that generates two ‘piles’ of strata. Each pile will therefore randomly contain one of the first two strata, one of the second two, and so on. Thus, each pile will contain one sample from each of the strata that would have been generated if only N\textsubscript{s}/2 total samples were to be taken. The full sampling sequence is generated by randomly shuffling each pile and then concatenating the two sequences.

For four binary subdivisions the same approach is extended, with the first flipper passing its ‘left’ and ‘right’ outputs to two lower-level flippers. This results in four ‘piles’ of strata, which again are just randomly shuffled and then concatenated. The same approach is simply extended to generate eight or sixteen strata where possible.

For risk analyses, it is frequently necessary to evaluate the low-probability, high-consequence end of the distribution of the performance of the system. Because the models for such systems are often complex (and hence need significant computer time to simulate), and it can be difficult to use the conventional Monte Carlo approach to evaluate these low-probability, high-consequence outcomes, as this may require excessive numbers of realizations.

To facilitate these type of analyses, GoldSim allows you to utilize an importance sampling algorithm to modify the conventional Monte Carlo approach so that the high-consequence, low-probability outcomes are sampled with an enhanced frequency. During the analysis of the results which are generated, the biasing effects of the importance sampling are reversed. The result is high-resolution development of the high-consequence, low-probability "tails" of the consequences, without paying a high computational price.

The following elements permit importance sampling:

- Stochastic
- Random Choice
- Timed Event Generator
- Action (in the Reliability Module)
- Function (in the Reliability Module)

**Note:** Importance sampling is only applied to the first sampled random value in each realization for elements with importance sampling enabled. Subsequent samples will use random sampling.

**Note:** In addition to the importance sampling method described here (in which you can choose to force importance sampling on either the low end or high end of a Stochastic element’s range), GoldSim also provides an advanced feature that supports custom importance sampling that can be applied over user-defined regions of the Stochastic element’s range.
Read more: Customized Importance Sampling Using User-Defined Realization Weights (page 1013).

**Warning:** Importance sampling affects the basic Monte Carlo mechanism, and it should be used with great care and only by expert users. In general, it is recommended that only one or at most a very few parameters should use importance sampling, and these should be selected based on sensitivity analyses using normal Monte Carlo sampling. **Importance sampling should only be used for elements whose distribution tails will not be adequately sampled by the selected number of realizations.**

### How Importance Sampling Works

Importance sampling is a general approach to selectively enhance sampling of important outcomes for a model. In principle, the approach is simple:

1. Identify an important subset of the sampling space;
2. Sample that subset at an enhanced rate; and
3. When analyzing results, assign each sample a weight inversely proportional to its enhancement-factor.

In conventional Monte Carlo sampling (with or without Latin Hypercube), each realization is assumed equally probable. It is straightforward, however, to incorporate a weight associated with each sample in order to represent the relative probability of the sample compared to the others.

The conventional Monte Carlo approach is as shown below. A uniform 0 – 1 random variable \( u \) is sampled, and its value is then used as input to the inverse cumulative distribution function of the random variable:

![Random Function](image)

In order to do importance sampling, the original uniformly-distributed random numbers are first mapped onto a non-uniform ‘biased’ sampling function \( s \):

![Biased Variate](image)
The biased variate $s$ is then used to generate the random function. Since the input random numbers are no longer uniformly distributed, the resulting sample set is selectively enriched in high-consequence results:

In general, any continuous monotonic biasing function $s$ which spans the range 0-1, and has $s(0) = 0$ and $s(1) = 1$ can be used to generate the set of input random numbers. The weight associated with each sampled realization is $ds/du$, the slope of the biasing function $s$ at the sampled point.

When a number of independent random variables are involved in a model, then the weight associated with a given realization is simply equal to the product of the weights of all parameters.

GoldSim uses simple functions to selectively enhance either the upper or the lower end of an element’s probability distribution.

The biasing function for enhancing the lower end of a distribution is:

$$s = u - \frac{u}{1 + au} + \frac{u^2}{1 + a}$$

where $a$ is a function of the number of elements that are using importance sampling. This is equal to zero if only one element uses importance sampling, and is equal to ten times the number of importance sampled elements in all other cases. The effect of increasing $a$ is to restrict the importance sampling to a smaller subset of the full range of the random value, which reduces the negative impacts of importance sampling numerous variables in the same model.

The sample weight is given by:

$$w = \frac{ds}{du} = 1 - \frac{1}{(1 + au)^2} + \frac{2u}{1 + a}$$

**Note:** For the first 10,000 realizations where GoldSim uses the expected values of the LHS strata the weight given to each sample is equal to the integral of $s$ over the stratum divided by the corresponding integral of $u$ over the strata. For the 10,001$^{st}$ and subsequent realizations GoldSim will calculate $s$ and $w$ for the sampled point.

The biasing function for enhancing the upper end is:

$$s_{upper} = 1 - s_{lower} = 1 - \left(1 - \left(1 - \frac{(1 - u)}{1 + a(1 - u)} + \frac{(1 + u)^2}{1 + a}\right)\right)$$
and the corresponding sample weight is given by:

\[ w_{\text{upper}} = w_{\text{lower}} (1-u) = 1 - \frac{2(1-u)}{(1 + a(1-u))^2} \]

The following plot shows the upper and lower biasing function when a single element utilizes importance sampling (an \( a \) value of zero):

![Importance Sampling Algorithm, a=0](image)

The following figure shows the bias function when three elements are importance sampled (an \( a \) value of 30):

![Importance Sampling Algorithm, a=30](image)

Note the less prominent bias as the number of importance sampled elements increases.
The Stochastic element provides the option to choose between upper and lower end enhancement when importance sampling is enabled. However, the other elements that utilize importance sampling (the Timed Event element, the Reliability elements and the Random Choice element) importance sample only one end of the distribution.

Timed Events will use lower end importance sampling on the time to event distribution specified by the user. The Random Choice element has a slightly different behavior. When importance sampling is enabled, the Random Choice element sorts the probability of each outcome from lowest probability to highest probability and assigns them to sections of a uniform distribution. This uniform distribution is then importance sampled at the lower end, so that the least probable outcomes are enhanced.

The Reliability elements will use a combination of these approaches. Time based failure modes behave in a similar manner to the timed event elements. Modes with a “probability of failure” perform importance sampling to enhance the number of failure outcomes.

It is important to note that only the first sampled value utilizes importance sampling. This means that only the first event from a Timed Event, the first Random Choice and the first occurrence of each Failure Mode will use importance sampling.

Representing Random (Poisson) Events

Timed Event elements can be specified to produce discrete event signals regularly or randomly.

Read more: Timed Event Elements (page 365).

Random events are simulated as occurring according to a Poisson process with a specified rate of occurrence. If an event occurs according to a Poisson process, the probability that N events will occur during a time interval T is described by the following expression (Cox and Miller, 1965):

$$P(N) = \frac{e^{-\lambda T} (\lambda T)^N}{N!}$$

where:

- $P(N)$ is the probability of N occurrences of the event within the time interval T;
- $T$ is the time interval of interest;
- $\lambda$ is the annual rate of occurrence; and
- $N$ is the number of occurrences of the event within the time interval T.

The expected (mean) number of occurrences during the time interval is equal to $\lambda T$.

The Poisson distribution also has the property that the intervals between events are exponentially distributed (Benjamin and Cornell, 1970):

$$F(t) = 1 - e^{-\lambda t}$$

where $F(t)$ is the probability that the time to the next event will be less than or equal to $t$. 

If you indicate that the event can only occur once, GoldSim simulates the first occurrence according to the above equations, but does not allow the event to occur again.

Note that the rate of occurrence can be specified to be a function of time (i.e., the event process can be non-stationary).

**Computing and Displaying Result Distributions**

Probabilistic results may be viewed in the form of a CDF (or a CCDF). This section describes how the values realized during the simulations are used to generate the CDF (or CCDF) seen by the user.

Within GoldSim Monte Carlo results are stored in a particular data structure, referred to here as the *results array*. As the simulation progresses, each specific Monte Carlo realization result is added to the results array as a pair of values; the value realized and the weight given by GoldSim to the value. The array is filled "on the fly", as each new realization is generated. Theoretically, each separate realization would represent a separate entry in the results array (consisting of a value and a weight). If unbiased sampling were carried out each separate entry would have equal weight.

As implemented in GoldSim, however, the number of data pairs in the results array may be less than the number of realizations: There are two reasons why this may be the case:

- If multiple results have identical values, there is no need to have identical data pairs in the results array: the weight associated with the particular value is simply adjusted (e.g., if the value occurred twice, its weight would be doubled).
- For computational reasons, the results array has a maximum number of unique results which it can store. The maximum number for post-processing GoldSim simulation results is 25,000. If the number of realizations exceeds these limits, results are "merged" in a self-consistent manner. The process of merging results when the number of realizations exceeds 25,000 is discussed below.

To merge a new result with the existing results (in cases where the number of realizations exceeds one of the maxima specified above), GoldSim carries out the following operations:

- GoldSim finds the surrounding pair of existing results, and selects one of them to merge with. GoldSim selects this result based on the ratio of the distance to the result to the weight of the result (i.e., the program preferentially merges with closer, lower weight results).
- After selecting the result to merge with, GoldSim replaces its value with the weighted average of its existing value and the new value; it then replaces its weight with the sum of the existing and new weights.

There is one important exception to the merging algorithm discussed above: If the new result will be an extremum (i.e., a highest or a lowest), GoldSim replaces the existing extremum with the new one, and then merges the existing result instead. This means that GoldSim never merges data with an extremum.

**Plotting a CDF**

Plotting the CDF from the results array is straightforward. The basic algorithm assumes that the probability distribution between each adjacent pair of result values is uniform, with a total probability equal to half the sum of the weights of
Computing and Displaying Result Distributions

the values. One implication of this assumption is that for a continuous distribution the probability of being less than the smallest value is simply equal to half the weight of the lowest value and the probability of being greater than the highest value is half the weight of the highest value.

For example, if we have ten equally weighted results in a continuous distribution, there is a uniform probability, equal to 0.1, of being between any two values. The probability of being below the lowest value or above the highest value would be 0.05. GoldSim extrapolates the value at a probability level of 0 using the slope between the first two observations. Similarly the slope between the last two observations is used to estimate the value at a probability level of 1.

Note: Extrapolation is not carried out for Milestone result distributions, Reliability element failure and repair time distributions, Decision Analysis results, and for Sampled Stochastic distributions if the option to extrapolate is not checked.

In certain circumstances there are several minor variations to the basic algorithm discussed above:

- If a large number of results are identical, GoldSim assumes the entire distribution is discrete (rather than continuous), and lumps the probabilities at the actual values sampled. In particular, if more than 50% of the realization results were identical to an existing result (and there are less than 1000 results), GoldSim presumes the distribution is discrete and plots it accordingly. The user can observe this by sampling from a binomial distribution.

- GoldSim uses a heuristic algorithm to decide if each specific result represents a discrete value: if any result is repeated, GoldSim treats the result as a discrete value and does not ‘smear’ it. For example, suppose the result is 0.0 30% of the time, and normal (mean=10, s.d.=2) the rest of the time. The first result value would be 0.0, with a weight of about 0.3. The second value would be close to 8, with a weight of 1/# realizations. We would not want to smear half of the 0 result over the range from 0 to 8!

When the user selects the confidence bounds options (discussed below), a different algorithm is used to display and plot CDF values. In particular, the displayed value is simply the calculated median (50th percentile) in the probability distribution for the “true” value of the desired quantile.

Displaying PDFs is much more difficult than CDFs, as unless a large number of realizations are run, PDFs tend to be “noisy” even if the corresponding CDF appears smooth (small changes in the slope of the CDF are typically not noticeable to the eye, but these can translate into very noticeable “spikes” in the PDF).

To display a PDF, GoldSim creates a series of equally-spaced bins, with a constant density value within each bin. The density within each bin is the average slope of the CDF over the bin. The number of bins automatically increases with the number of realizations. In particular, the number of bins is equal to the square root of the number of results being considered. The maximum number of bins used is 1000 (and the minimum is 1).
GoldSim is able to perform a statistical analysis of Monte Carlo results to produce confidence bounds on the mean of a probabilistic result. These bounds reflect uncertainty in the probability distribution due to the finite number of Monte Carlo realizations - as the number of realizations is increased, the limits become narrower. The confidence bounds on the mean are displayed in the Statistics section of the Distribution Summary dialog when viewing Distribution Results if the Confidence Bounds checkbox is checked.

Read more: Viewing a Distribution Summary (page 643).

This approach to compute the confidence bounds uses the t distribution, which is strictly valid only if the underlying distribution is normal. The 5% and 95% confidence bounds on the population mean are calculated as defined below:

\[ P\left( \bar{X} + t_{0.05} \frac{s_x}{\sqrt{n}} \right) < \mu = 0.05 \]

and

\[ P\left( \bar{X} + t_{0.95} \frac{s_x}{\sqrt{n}} \right) \geq \mu = 0.05 \]

where:
- \( \bar{X} \) is the sample mean
- \( t_{0.05} \) is the 5% value of the t distribution for \( n-1 \) degrees of freedom
- \( t_{0.95} \) is the 95% value, \( = -t_{0.05} \)
- \( s_x \) is the sample standard deviation
- \( \mu \) is the true mean of the population, and
- \( n \) is the number of samples (realizations).

As the number of realizations, \( n \), becomes large, the Central Limit theorem becomes effective, the t distribution approaches the normal distribution, and the assumption of normality is no longer required. This may generally be assumed to occur for \( n \) in the order of 30 to 100 realizations even for results of highly-skewed distributions.

GoldSim is able to perform a statistical analysis of Monte Carlo results to produce confidence bounds on the resulting probability distribution curves. These bounds reflect uncertainty in the probability distribution due to the finite number of Monte Carlo realizations - as the number of realizations is increased, the limits become narrower. The confidence bounds are displayed when viewing Distribution Results if the Show Confidence Bounds button in the Distribution display window is pressed.

The confidence bounds appear as different-colored curves on the probability plots produced by the GoldSim user interface. The bounds represent 5% and 95% confidence limits on the distribution value at each probability level. Confidence bounds cannot be displayed for PDFs.

The theory used to produce the confidence bounds has several limitations:

- The bounds on distributions can only be calculated for cases where all realizations have equal weights. If importance sampling is used for any parameter is used, GoldSim will not display confidence bounds.
- The confidence bounds cannot be calculated for values less than the smallest result, or greater than the largest result. As a result of this, the
confidence-bound curves do not generally reach all of the way to the tails of the result plots.

In cases with relatively few stochastic parameters, Latin Hypercube sampling can increase the accuracy of the probability distributions. The confidence bounds are not able to reflect this improvement, and as a result will be conservatively wide in such cases.

Suppose we have calculated and sorted in ascending order \( n \) random results \( r_i \) from a distribution. What can we say about the \( q^{th} \) quantile \( x_q \) (e.g., \( q=0.9 \)) of the underlying distribution?

Each random result had a probability of \( q \) that its value would be less than or equal to the actual \( q^{th} \) quantile \( x_q \). The total number of results less than \( x_q \) was therefore random and binomially distributed, with the likelihood of exactly \( i \) results \( \leq x_q \) being:

\[
P(i) = P(r_i \leq x_q \leq r_{i+1}) = \binom{n}{i} q^i (1-q)^{n-i} = \frac{n!}{i!(n-i)!} q^i (1-q)^{n-i}
\]

Note that there may be a finite probability that the value of \( x_q \) is less than the first or greater than the largest result: for example, if 100 realizations \( r_i \) were sampled, there would be a probability of 0.366 that the 0.99 quantile exceeded the largest result. The 100 realization probability distribution for \( x_{0.99} \) is as follows:

<table>
<thead>
<tr>
<th>Between Results</th>
<th>Probability</th>
<th>Cumulative Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt;94)</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>94 and 95</td>
<td>0.0005</td>
<td>0.0005</td>
</tr>
<tr>
<td>95 and 96</td>
<td>0.003</td>
<td>0.0035</td>
</tr>
<tr>
<td>96 and 97</td>
<td>0.015</td>
<td>0.0185</td>
</tr>
<tr>
<td>97 and 98</td>
<td>0.061</td>
<td>0.0795</td>
</tr>
<tr>
<td>98 and 99</td>
<td>0.1849</td>
<td>0.2644</td>
</tr>
<tr>
<td>99 and 100</td>
<td>0.370</td>
<td>0.6344</td>
</tr>
<tr>
<td>(&gt;100)</td>
<td>0.366</td>
<td>1.</td>
</tr>
</tbody>
</table>

GoldSim assumes that the probability defined by the equation presented above is uniformly distributed over the range from \( r_i \) to \( r_{i+1} \), and interpolates into the Monte Carlo results-list to find the 5% and 95% cumulative probability levels for \( x_q \). For example, for 100 realizations, the 5% confidence bound on the 0.9 quantile is 0.31 of the distance from result 85 to result 86, and the 95% confidence bound is 0.22 of the distance from result 95 to result 96.

<table>
<thead>
<tr>
<th>Between Results</th>
<th>Probability</th>
<th>Cumulative Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt;85)</td>
<td>-</td>
<td>0.040</td>
</tr>
<tr>
<td>85 and 86</td>
<td>0.033</td>
<td>0.073</td>
</tr>
<tr>
<td>86 and 87</td>
<td>0.051</td>
<td>0.124</td>
</tr>
<tr>
<td>87 and 88</td>
<td>0.074</td>
<td>0.198</td>
</tr>
<tr>
<td>88 and 89</td>
<td>0.099</td>
<td>0.297</td>
</tr>
</tbody>
</table>
Using the above probability distribution, it is also possible to calculate the expected value of \( x_q \). This approach appears (by experimentation) to provide a slightly more reliable estimate of \( x_q \) than the conventional Monte Carlo approach of directly interpolating into the results list. The expected value of \( x_q \) is calculated by summing the product of the probability of \( x_q \) lying between each pair of results and the average of the corresponding pair of result values, i.e.,

\[
\hat{x}_q = \sum_{i=1}^{n} \left[ i \cdot \frac{r_{i+1} + r_i}{2} \right] \cdot P(i)
\]

When using this equation to estimate a very high or low quantile, a problem arises when the probability level, \( q \), is near to 0 or 1, as there can be a significant probability that \( x_q \) lies outside the range of results. In the first table presented above, for example, there is a 0.366 chance of \( q_{0.99} \) exceeding the largest result. In such cases, an estimate of the expected value of \( x_q \) can be found by extrapolating from the probabilities within the range of the results. Obviously, however there are limits to extrapolation and without knowledge of the actual distributional form no extrapolation would produce a reliable estimate of (say) the 0.9999 quantile if only 100 realizations had been performed.

In evaluating the binomial distribution for large values of \( n \), large numbers can be generated which can cause numerical difficulties. To avoid these difficulties, when the number of realizations (\( n \)) is greater than 100, GoldSim uses either the normal or Poisson approximations to the binomial distribution. The Poisson approximation is used when \( i \) or \( n - i \) is less than 20 and the normal distribution is used otherwise. These approximations are described in any introductory statistics text.

The Conditional Tail Expectation (CTE) is the expected value of the result given that it lies above a specified value or quantile (Hardy, 2006). The CTE is displayed in the ‘Calculator’ portions of the Stochastic and Result Distribution elements, and is an optional output type for a Monte Carlo SubModel element.

Read more: Specifying the Distribution for a Stochastic Element (page 155); Viewing a Distribution Summary (page 643); Creating the Output Interface to a SubModel (page 984).

For a tail starting at value \( k \), the CTE is defined as:
Computing Sensitivity Analysis Measures

\[ CTE_k = \frac{1}{1 - F(k)} \int_k^\infty x \cdot f(x) \cdot dx \]

where:

- \( F(k) \) is the cumulative probability of value \( k \)
- \( f(x) \) is the probability density at value \( x \)

The CTE is related to another statistical measure of a distribution, the partial expectation (PE). The PE is defined as:

\[ PE_k = \int_k^\infty (x - k) \cdot f(x) \cdot dx \]

The CTE and PE are related as follows:

\[ PE_k = (1 - F(k)) \left[ CTE_k - k \right] \]

\[ CTE_k = \frac{PE_k}{(1 - F(k))} + k \]

Computing Sensitivity Analysis Measures

GoldSim provides a number of statistical sensitivity analyses through the multi-variate result display option. If you press the Sensitivity Analysis… button from the Multi-Variate Result dialog, a table such as this is displayed:

```
<table>
<thead>
<tr>
<th>Result</th>
<th>Importance Measure</th>
<th>Correlation Coefficient</th>
<th>Regression Coefficient</th>
<th>Partial Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.384</td>
<td>0.653</td>
<td>0.682</td>
<td>0.528</td>
</tr>
<tr>
<td>2</td>
<td>0.186</td>
<td>0.178</td>
<td>0.408</td>
<td>0.827</td>
</tr>
<tr>
<td>3</td>
<td>0.264</td>
<td>0.456</td>
<td>0.544</td>
<td>0.000</td>
</tr>
<tr>
<td>4</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>
```

This table displays measures of the sensitivity of the selected result (the output from which you selected Multi-Variate Result…) to the selected input variables.

**Note:** You can control whether the sensitivity analyses are based on the values or the ranks of the variables and the result via the Use Ranks button at the top of the display.

The measures that GoldSim computes are:

- Coefficient of determination;
- Correlation coefficient;
- Standardized regression coefficient (SRC);
- Partial correlation coefficient; and
- Importance measure.
Computing the Coefficient of Determination

The coefficient of determination represents the fraction of the total variance in the result that can be explained based on a linear (regression) relationship to the input variables (i.e., Result = aX + bY + cZ + ...). The closer this value is to 1, the better that the relationship between the result and the variables can be explained with a linear model.

The formulation for the coefficient of determination (R²) can be found in Iman (1985). Note that if all of the variables on uncorrelated, then:

\[ R^2 = \sum \frac{C_i}{n} \]

where \( C_i \) is the correlation coefficient for variable i.

Computing Correlation Coefficients

Rank (Spearman) or value (Pearson) correlation coefficients range between -1 and +1, and express the extent to which there is a linear relationship between the selected result and an input variable.

The value correlation coefficient is computed as follows:

\[ C_{rp} = \frac{\sum_{i=1}^{n} (p_i - m_p)(r_i - m_r)}{\sqrt{\sum_{i=1}^{n} (p_i - m_p)^2 \sum_{i=1}^{n} (r_i - m_r)^2}} \]

where:

- \( C_{rp} \) = the value correlation coefficient;
- \( n \) = the number of selected data points (realizations);
- \( p_i \) = value of output \( p \) for realization \( i \);
- \( r_i \) = value of output \( r \) for realization \( i \);
- \( m_p \) = mean value of output \( p \); and
- \( m_r \) = mean value of output \( r \).

Note that the value correlation coefficient as defined here provides a measure of the linear relationship between two variables.

Furthermore, it may be affected by a few aberrant pairs (i.e., a good alignment of a few extreme pairs can dramatically improve an otherwise poor correlation coefficient; likewise, an otherwise good correlation could be ruined by the poor alignment of a few extreme pairs).

To overcome these problems, the value correlation coefficient can be supplemented by the rank correlation coefficient.

The rank correlation coefficient (also referred to as the Spearman rank correlation coefficient) is computed using the same equation as that of the value correlation coefficient with the ranks of the data values being used rather than the actual values:

\[ C_{rp,\text{rank}} = \frac{\sum_{i=1}^{n} (R_{p_i} - m_{R_p})(R_{r_i} - m_{R_r})}{\sqrt{\sum_{i=1}^{n} (R_{p_i} - m_{R_p})^2 \sum_{i=1}^{n} (R_{r_i} - m_{R_r})^2}} \]
where:

\[ C_{rp,rank} = \text{the rank correlation coefficient}; \]
\[ n = \text{the number of selected data points (realizations)}; \]
\[ R_p = \text{the rank (from 1 to n) of output p for realization I}; \]
\[ R_r = \text{the rank (from 1 to n) of output p for realization I}; \]
\[ m_{Rp} = \text{mean value of the rank of output p}; \]
\[ m_{Rr} = \text{mean value of the rank of output r}. \]

In GoldSim, the ranks of equal values are the same. For example, if the lowest two realizations of an output were the same, their ranks would both be 1.5 (the mean of 1 and 2), with the third lowest value being assigned a rank of 3.

**Note:** A correlation coefficient cannot be computed for a pair of outputs if one of the outputs has a standard deviation of zero (i.e., is constant). In this case, GoldSim sets the correlation coefficient to zero.

### Computing Standardized Regression Coefficients (SRC)

Standardized regression coefficients range between -1 and +1 and provide a normalized measure of the linear relationship between variables and the result. They are the regression coefficients found when all of the variables (and the result) are transformed and expressed in terms of the number of standard deviations away from their mean. GoldSim’s formulation is based on Iman et al (1985).

The use of standardized regression coefficients to identify the importance of correlated input variables is described in Iman et al (1985) and Mishra (2004). In this approach the correlation matrix \( C \) for the input variables is augmented, with an additional row/column assigned for the result (GoldSim uses the first row/column for this purpose).

The standardized regression coefficients are based on the inverse of the augmented correlation matrix, and are found by dividing the terms in the augmented column of the matrix by the negative of the augmented diagonal term:

\[
SRC_{y,i} = \frac{-c_{iy}}{c_{yy}}
\]

where:

\[ SRC_{y,i} = \text{the standardized regression coefficient of the result (Y) with respect to input variable X}_i; \]
\[ c_{iy} = \text{the off-diagonal term in the inverted correlation matrix for row i, column y}; \]
\[ c_{yy} = \text{the diagonal term in the inverted correlation matrix corresponding to the result y}. \]

### Computing Partial Correlation Coefficients

Partial correlation coefficients reflect the extent to which there is a linear relationship between the selected result and an input variable, after removing the effects of any linear relationships between the other input variables and both the result and the input variable in question. For systems where some of the input variables may be correlated, the partial correlation coefficients represent the “unique” contribution of each input to the result. GoldSim’s formulation is based on Iman et al (1985).
The partial correlation coefficient $P_{y,i}$ is calculated as:

$$P_{y,i} = -\frac{c_{iy}}{\sqrt{c_{ii}c_{yy}}}$$

where:

$P_{y,i} = \text{the partial correlation coefficient of the result (Y) to input variable } X_i$;

$c_{iy} = \text{the off-diagonal term in the inverted correlation matrix for row } i, \text{ column } y$; and

$c_{ii}, c_{yy} = \text{the diagonal terms in the inverted correlation matrix corresponding to the input variable and the result, respectively.}$

Note that if any two or more of the input variables are linearly dependent, for example if they are perfectly correlated, then the correlation matrix becomes singular and cannot be inverted. GoldSim, which uses Choleski decomposition to compute the inverse, will automatically adjust the correlation matrix if necessary in order to compute the partial correlation coefficients. This adjustment, which takes the form of ‘weakening’ of the off-diagonal terms, does not affect the displayed correlation coefficients.

If there is a nonlinear, non-monotonic relationship between an input variable and the result, conventional correlation coefficients may not reveal the relationship. The Importance Measure computed by GoldSim is a normalized version of the measure $E[V(Y|X_i)]$ discussed in Saltelli and Tarantola (2002). The Saltelli and Tarantola measure represents the expected value of the variance if the input variable $X_i$ was not uncertain. Thus, the smaller this value, the more the input variable controls the result.

The GoldSim version of this measure is normalized as follows:

$$M_{y,i} = 1 - \frac{E[V(Y|X_i)]}{V_y}$$

where:

$M_{y} = \text{the GoldSim importance measure for the sensitivity of the result (Y) to input variable } X_i$;

$V_y = \text{the current variance in the result Y}$; and

$E[V_y(Y|X_i)] = \text{the expected value of } V_y \text{ if the input variable } X_i \text{ was perfectly known.}$

Thus, GoldSim’s Importance Measure $M_{y,i}$ represents the fraction of the result variance that is explained by $X_i$. Note that, like the correlation coefficients and scatter-plots, the importance measure can be calculated using either values or ranks, as specified in the main property window for the multivariate element.

GoldSim calculates $M_{y,i}$ numerically, using the following method. Construct a 2-d scatter plot of the results, with the $X_i$ values on the horizontal axis and the result on the vertical axis. If $X_i$ is very important to the result, then this plot will show a clustering around a central line or curve:
Subdivide the plot into \( N_s \) vertical segments based on the ranks of the \( X_i \)-values, where \( N_s \) equals the square root of the number of model realizations. For each of the segments, estimate the variance of \( Y \) within that segment by using a weighting function that assigns decreasing weights to the results as their distance from the center of the segment increases. Then compute the average value of the variance over all of the segments, i.e. \( E[V_Y (Y|X_i)] \). For the weighting function, GoldSim uses the density of a beta distribution having a mean equal to the \( X \)-value at the center of the segment, a standard deviation equal to the segment width, and upper and lower bounds corresponding to the range of \( X \)-values.

The example plot above is based on calculating:

\[
Y = X_1 + X_2^2 + X_3^3
\]

where:

- \( X_1 \) is a random variable with a \( U(1,2) \) distribution;
- \( X_2 \) is a random variable with a \( U(-10,10) \) distribution; and
- \( X_3 \) is a random variable with a \( U(-3,3) \) distribution.

The plot shown above, plotting \( Y \) vs \( X_2 \), clearly reveals the importance of \( X_2 \) in this model. Conventional correlation analysis is completely unable to recognize this sensitivity, as indicated by the correlation coefficient of effectively zero (-0.015) in the screen-shot below. However, the importance measure is sensitive to it, and identifies \( X_2 \) as the most importance variable.
References

The references cited in this appendix are listed below.


Appendix C: Implementing External (DLL) Elements

Begin at the beginning and go on till you come to the end; then stop.

Lewis Carroll, Alice in Wonderland

Appendix Overview

GoldSim allows you to develop separate program modules (written in C, C++, Pascal, FORTRAN or other compatible programming languages) which can then be directly coupled with the main GoldSim algorithms. These user-defined modules are referred to as external functions, and are linked into GoldSim as DLLs (Dynamic Link Libraries) at run time. GoldSim interfaces with the DLL via an External element.

Read more: External (DLL) Elements (page 931).

Integrating your external program module into GoldSim requires that you develop a "wrapper" or "shell" around the function and compile it into a DLL. This appendix discusses the details of how external functions must be coded and compiled.

In this Appendix

This appendix discusses the following:

• Understanding External (DLL) Elements
• Implementing an External Function
• External Function Examples
• External Function Calling Sequence
• DLL Calling Details
Understanding External (DLL) Elements

External functions work with External elements to do calculations or other manipulations that are not included in the standard capabilities of GoldSim. The external function facility allows special purpose calculations or manipulations to be accomplished with more flexibility, speed or complexity than with the standard GoldSim element types.

The external functions are bound to the GoldSim executable code at run time using DLL technology. The DLL files should be present in the same folder as the GoldSim .gsm file, in the same folder as the GoldSim executable file, or elsewhere in the user’s path.

Note that these functions are external to GoldSim and are not covered by the standard GoldSim verification process. The user is responsible for any necessary testing of external functions.

Every external function is called by GoldSim with specific requests. The requests include initialization, returning the function version number, performing a normal calculation, and "cleaning up" after a simulation. The function name and argument list (the set of input and output data for the function) are specified by the GoldSim user when setting up the External element.

External functions should provide their own error handling, message handling, file management and memory management if required. It is essential that when it receives a “clean up” request, an external function should release any dynamically acquired memory and close any open files.

Note: In the case of an error condition, the external function should always return an error code to GoldSim, so that the user can be informed about the problem and the simulation can be terminated cleanly with no memory leaks.

Implementing an External Function

The implementation of the external function is left to the programmer, but several restrictions apply so that the functions can be correctly called by GoldSim. They are:

- The function return value is ignored. For example, use void functions in C/C++, or subroutines in FORTRAN.
- Data are passed from GoldSim to the external function and back again to GoldSim via arrays of double precision floating point input and output arguments.
- Input arguments must not be modified by the external function. Doing so may cause memory corruption in the DLL and/or GoldSim.
- Each function must manage its own initialization and memory allocation, and its own messages to the user (if any).
- Unique external function (or subroutine) names must be defined in each DLL. In C++, function names are case-sensitive, while in FORTRAN, the case of the external subroutine name is determined from the DLL.
build options. In all instances, the function name specified in GoldSim is case-sensitive, and must agree with the case specified in the DLL.

- All files required to run your specific DLL must be properly installed on the machine where GoldSim is running. This includes any additional runtime libraries required by your DLL.

- Most development environments allow both static and dynamic binding of runtime libraries to DLLs. DLLs built with static binding are stand-alone, with all run-time library references included in the DLL file. However, if a DLL is built with dynamic binding, the runtime libraries are not included. For a DLL with dynamic binding, the runtime libraries (e.g. msvcrtd.dll for Visual C++ or libifcoremd.dll for Intel Visual Fortran) must be present and in the environment PATH variable on the machine where GoldSim is running.

When calling methods from the DLL, GoldSim always expects the following C/C++ function signature:

```c
extern "C" void __declspec(dllexport)
MyExternalFcn(int XFMethod,
int* XFState,
double* inargs,
double* outargs)
```

The extern "C" specifies C language style linkage between GoldSim and the DLL, and __declspec(dllexport) makes the function visible outside the DLL.

For Intel Visual Fortran, the following subroutine signature can be used:

```fortran
subroutine my_external_fcn(xfmethod, xfstate, inargs, outargs)
  !DEC$ ATTRIBUTES dllexport,c :: add_mult_scalars
  !DEC$ ATTRIBUTES value :: nmethod
  !DEC$ ATTRIBUTES reference :: nstatus
  !DEC$ ATTRIBUTES reference :: dinputs
  !DEC$ ATTRIBUTES reference :: doutputs
```

Other FORTRAN development environments may require different attributes for the proper linkage to GoldSim.

**Note:** Arrays in C/C++ start at zero, rather than one (the default for FORTRAN). Array indices in this section use the C/C++ convention.

The arguments are:

```plaintext
int XFMethod;   // Action that the external function
               // must perform (see table below)
int* XFState;  // Returned value success 0 or fatal 1
double* inargs; // Array of input arguments
double* outargs; // This array returns different
                 // information when different XFmethod
                 // values are passed to the external
```
// function.

The values for XFmethod are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>XF_INITIALIZE                  Initialize (called at the beginning of each</td>
</tr>
<tr>
<td></td>
<td>realization).</td>
</tr>
<tr>
<td></td>
<td>No arguments are passed on this call.</td>
</tr>
<tr>
<td>1</td>
<td>XF_CALCULATION                 Normal calculation.</td>
</tr>
<tr>
<td></td>
<td>Total number of output arguments are returned on this call.</td>
</tr>
<tr>
<td></td>
<td>outargs[0] = 1st result from the function</td>
</tr>
<tr>
<td></td>
<td>outargs[1] = 2nd result from the function etc.</td>
</tr>
<tr>
<td>2</td>
<td>XF_REP_VERSION                  External functions report their versions.</td>
</tr>
<tr>
<td></td>
<td>No input arguments are passed on this call.</td>
</tr>
<tr>
<td></td>
<td>outargs[0] = external function version, e.g. 1.23</td>
</tr>
<tr>
<td>3</td>
<td>XF_REP_ARGUMENTS                External functions report the # of input</td>
</tr>
<tr>
<td></td>
<td>and output arguments.</td>
</tr>
<tr>
<td></td>
<td>No input arguments are passed on this call.</td>
</tr>
<tr>
<td></td>
<td>outargs[0] = # of input arguments.</td>
</tr>
<tr>
<td>99</td>
<td>XF_CLEANUP                      Close any open files, and optionally release dynamic memory at the end of a simulation.</td>
</tr>
<tr>
<td></td>
<td>No arguments are passed on this call.</td>
</tr>
</tbody>
</table>

The returned values for XFState are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OK, continue GoldSim</td>
</tr>
<tr>
<td>&gt;0 and &lt; 99</td>
<td>terminate GoldSim</td>
</tr>
<tr>
<td>99</td>
<td>OK, unload the DLL</td>
</tr>
</tbody>
</table>

The following two return codes may only be used for an XF_CALCULATION call:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Fatal error, error message pointer returned</td>
</tr>
<tr>
<td>-2</td>
<td>More result memory required; the total amount (in doubles)</td>
</tr>
<tr>
<td></td>
<td>required is returned in outargs[0]. This may be required of the</td>
</tr>
<tr>
<td></td>
<td>external function is returning a table or time series definition.</td>
</tr>
<tr>
<td></td>
<td>(Note that the additional memory is retained until the end of the simulation.)</td>
</tr>
</tbody>
</table>

The memory for the inargs and outargs arrays is allocated by GoldSim at the start of the simulation, based upon the inputs and outputs specified in the Interface tab of the External element properties dialog. The number of input and output arguments is verified during the XF_REP_ARGUMENTS request. GoldSim counts up the number of input and output arguments it expects, and compares it to the numbers for each reported by the DLL.
Warning: It is the responsibility of the external function to ensure that it does not write to any output arguments beyond the number reported for XF_REP_ARGUMENTS. Doing so may cause memory corruption in GoldSim.

Argument Checking

GoldSim calculates the total number of input and output arguments by summing over the the inputs and outputs specified on the Interface tab of the External element properties dialog. Note that each scalar input or output counts as one argument, while array inputs or outputs count as the size of the array (rows * columns). The calculated totals are then compared to the external function’s reported number of input and output arguments. If the number of arguments defined by GoldSim does not agree with the number reported by the external function, GoldSim terminates with an error message.

However, note the following exceptions:

- If outargs[0] is returned as -1, the number of input arguments is not tested. This allows for functions where the DLL does not know in advance the number of input argument that it will receive.

- If the external function will be returning definitions for one or more Tables or Time Series (see below), GoldSim will not know in advance how long the Table definitions will be. In this case, the external function should specify a value for outargs[1] greater than or equal to the actual total number of arguments that may be returned. GoldSim will allocate this amount of memory for outargs. Note that this can be reset during the simulation by returning XFState=-2.

The Input and Output Argument Arrays

The content of input and output argument arrays is determined from the Interface tab of the External element properties dialog.

The following points should be noted:

- Input or outputs are processed in the order specified in the interface. All data from one input or output is contiguous, followed by the data from the next input or output.

- Scalar inputs and outputs are mapped to a single array argument.

- Vector input and output items are specified in order, one argument per item, according to the order specified in the array label.

- Matrix input and output items are specified item-by-item, with all items in one row together, followed by all items in the next row, and so on.

- Lookup Table definitions can be specified in the output interface, via a special format.

- Time Series definitions can be specified as inputs or outputs, also via a special format.

Lookup Table Definitions

External functions can also return Table definition elements to GoldSim. A table definition requires a specific sequence of values, depending whether it is a 1-D, 2-D, or 3-D table.

The sequence for a 1-D table is as follows:

- number of dimensions (in this case, 1)
- the number of rows
Implementing an External Function

- row value 1, row value 2, …, row value n
- dependent value 1, dependent value 2, …, dependent value n

The sequence for a 2-D table is as follows:
- number of dimensions (in this case, 2)
- the number of rows, the number of columns
- row value 1, row value 2, …, row value n
- column value 1, column value 2, …, column value n
- dependent(row 1, column 1), …, dependent(row 1, column n)
- dependent(row 2, column 1), …, dependent(row 2, column n)
- …
- dependent(row n, column 1), …, dependent(row n, column n)

**Warning:** This is different than the sequence used to read in an ASCII text file for a 2-D table.

The sequence for a 3-D table is as follows:
- number of dimensions (must be 3)
- the number of rows, the number of columns, the number of layers
- row value 1, row value 2, …, row value y
- column value 1, column value 2, …, column value x
- layer value 1, layer value 2, …, layer value z
- dependent(row 1, column 1, layer 1), …, dependent(row 1, column x, layer 1)
- dependent(row 2, column 1, layer 1), …, dependent(row 2, column x, layer 1)
- …
- dependent(row y, column 1, layer 1), …, dependent(row y, column x, layer 1)
- …
- dependent(row 1, column 1, layer z), …, dependent(row 1, column x, layer z)
- dependent(row 2, column 1, layer z), …, dependent(row 2, column x, layer z)
- …
- dependent(row y, column 1, layer z), …, dependent(row y, column x, layer z)
Warning: This is different than the sequence used to read in an ASCII text file for a 3-D table.

Time Series Definitions

External functions can also read and return Time Series Definition. A Time Series Definition consists of the following specific sequence of values.

1. The number 20 (this informs GoldSim that this is a Time Series)
2. The number -3 (this is a format number that informs GoldSim what version of the time series format is expected)
3. Calendar-based index: 0 if elapsed time; 1 if dates
4. An index (0,1,2,3) indicating what the data represents (0=instantaneous value, 1=constant value over the next time interval, 2=change over the next time interval, 3=discrete change)
5. The number of rows (0 for scalar time series)
6. The number of columns (0 for scalar and vector time series)
7. Number of series
8. For each series, the following is repeated:
   - The total number of time points in the series
   - Time point 1, Time point 2, ..., Time point n

The structure of the remainder of the file depends on whether the Time Series Definition represents a scalar, a vector, or a matrix.

For a scalar, the next sequence of values is as follows:
   - Value 1[time point 1], Value 2[time point 2], ..., Value[time point n]

For a vector, the next sequence of values is as follows:
   - Value[row1, time point 1], Value[row1, time point 2], ..., Value[row1, time point n]
   - Value[row2, time point 1], Value[row2, time point 2], ..., Value[row2, time point n]
   - ...
   - Value[rowr, time point 1], Value[rowr, time point 2], ..., Value[rowr, time point n]

For a matrix, the next sequence of values is as follows:
   - Value[row1, column1, time point 1], Value[row1, column1, time point 2], ..., Value[row1, column1, time point n]
   - Value[row1, column2, time point 1], Value[row1, column2, time point 2], ..., Value[row1, column2, time point n]
   - ...
   - Value[row1, columnnc, time point 1], Value[row1, columnnc, time point 2], ..., Value[row1, columnnc, time point n]
   - ...
External Function Examples

The following is a simple example implementation of DLL code that will work with an External element. This code takes two scalar elements as input, and returns the sum and product to GoldSim. For simplicity, this code is written in C, implemented with Visual C++. This external function can be used with the External.gsm example which can be found in the External subfolder of the General Examples folder in your GoldSim directory (accessed by selecting File | Open Example... from the main menu).

```c
// Global enumerations, useful for C-style implementations

// XFMethodID identifies the method types, used to identify the phase of the simulation
// that is currently in progress.
// XF_INITIALIZE - Called after DLL is loaded and before each realization.
// XF_CALCULATE - Called during the simulation, each time the inputs change.
// XF_REP_VERSION - Called after DLL load to report the external fcn version number.
// XF_REP_ARGUMENTS - Called after DLL load to report the number of input and output arguments.
// XF_CLEANUP - Called before the DLL is unloaded.

enum XFMethodID
{
    XF_INITIALIZE = 0, XF_CALCULATE = 1, XF_REP_VERSION = 2,
    XF_REP_ARGUMENTS = 3,
    XF_CLEANUP = 99
};

// XFStatusID identifies the return codes for external functions.

// XF_SUCCESS - Call completed successfully.
// XF_CLEANUP_NOW - Call was successful, but GoldSim should clean up and unload the DLL immediately.
// XF_FAILURE - Failure (no error information returned).
// XF_FAILURE_WITH_MSG - Failure, with DLL-supplied error message available. Address of error message is returned in the first element of the output arguments array.
// XF_INCREASE_MEMORY - Failed because the memory allocated for
```

- \( \text{Value[row\_r, column\_1, time\_point\_1]}, \text{Value[row\_r, column\_1, time\_point\_2]}, \ldots, \text{Value[row\_r, column\_1, time\_point\_n]} \)
- \( \text{Value[row\_r, column\_2, time\_point\_1]}, \text{Value[row\_r, column\_2, time\_point\_2]}, \ldots, \text{Value[row\_r, column\_2, time\_point\_n]} \)
- \( \ldots \)
- \( \text{Value[row\_r, column\_c, time\_point\_1]}, \text{Value[row\_r, column\_c, time\_point\_2]}, \ldots, \text{Value[row\_r, column\_c, time\_point\_n]} \)
// is too small. GoldSim will increase the size of the output argument array and try again.

enum XFStatusID
{
    XF_SUCCESS = 0, XF_FAILURE = 1, XF_CLEANUP_NOW = 99,
    XF_FAILURE_WITH_MSG = -1,
    XF_INCREASE_MEMORY = -2
};

extern "C" void __declspec(dllexport) AddMultScalarsInC(int     methodID,
                                                int*    status,
                                                double* inargs,
                                                double* outargs)
{
    *status = XF_SUCCESS;
    switch ( methodID )
    {
    case  XF_INITIALIZE:
        break;                                // nothing required

    case  XF_REP_VERSION:
        outargs[0] = 1.03;
        break;

    case  XF_REP_ARGUMENTS:
        outargs[0] = 2.0;                     // 2 scalar inputs expected
        outargs[1] = 2.0;                     // 2 scalar outputs
        break;

    case  XF_CALCULATE:
        outargs[0] = inargs[0] + inargs[1];   // return the sum
        outargs[1] = inargs[0]*inargs[1];     // return the product
        break;

    case  XF_CLEANUP:
        break;                                // No clean-up required

    default:
    *status = XF_FAILURE;
        break;
    }
}

The following code is the same algorithm, implemented in Intel Visual Fortran.
module gs_parameters
    implicit none

    ! Parameters to identify the method types, which indicate the phase of the
    ! simulation that is currently in progress.
    !
    ! INITIALIZE - Called after DLL is loaded and before each
    ! realization.
    ! CALCULATE - Called during the simulation, each time the
    ! inputs change.
    ! REPORT_VERSION - Called after DLL load to report the external fcn
    ! version number.
    ! REPORT_ARGUMENTS - Called after DLL load to report the number of
    ! input and output
    ! arguments.
    ! CLEANUP - Called before the DLL is unloaded.

    integer(4), parameter :: INITIALIZE       =  0
    integer(4), parameter :: CALCULATE        =  1
    integer(4), parameter :: REPORT_VERSION   =  2
    integer(4), parameter :: REPORT_ARGUMENTS =  3
    integer(4), parameter :: CLEANUP          = 99

    ! Parameters to identify the return codes for external functions.
    !
    ! SUCCESS - Call completed successfully.
    ! CLEANUP_NOW - Call was successful, but GoldSim should clean up
    ! and unload the DLL immediately.
    ! FAILURE - Failure (no error information returned).
    ! FAILURE_WITH_MSG - Failure, with DLL-supplied error message
    ! available.
    ! first element
    ! of the output arguments array.
    ! INCREASE_MEMORY - Failed because the memory allocated for output
    ! arguments
    ! is too small. GoldSim will increase the size of the
    ! output argument array and try again.

    integer(4), parameter :: SUCCESS          =  0
    integer(4), parameter :: FAILURE          =  1
    integer(4), parameter :: CLEANUP_NOW      = 99
    integer(4), parameter :: FAILURE_WITH_MSG =-1
    integer(4), parameter :: INCREASE_MEMORY  =-2

end module gs_parameters

! Utility module to specify the GoldSim parameter constants

subroutine add_mult_scalars(method_id, status, inargs, outargs)
    !DEC$ ATTRIBUTES dllexport,c :: add_mult_scalars
    !DEC$ ATTRIBUTES value       :: method_id
    !DEC$ ATTRIBUTES reference  :: status
    !DEC$ ATTRIBUTES reference  :: inargs
    !DEC$ ATTRIBUTES reference  :: outargs
use gs_parameters
implicit none
real(8), parameter :: VERSION  = 1.03
integer(4), parameter :: NINPUTS  = 2 ! Two scalar inputs expected
integer(4), parameter :: NOUTPUTS = 2 ! Two scalar outputs returned
integer(4) method_id, status
real(8)    inargs(*), outargs(*)

select case (method_id)
  case (INITIALIZE)
    status = SUCCESS
  case (REPORT_VERSION)
    outargs(1) = VERSION
    status = SUCCESS
  case (REPORT_ARGUMENTS)
    outargs(1) = NINPUTS
    outargs(2) = NOUTPUTS
    status = SUCCESS
  case (CALCULATE)
    outargs(1) = inargs(1) + inargs(2) ! return the sum
    outargs(2) = inargs(1)*inargs(2) ! return the product
    status = SUCCESS
  case (CLEANUP)
    status = SUCCESS
  case default
    status = FAILURE
end select

end subroutine add_mult_scalars

Additional DLL code examples can be found (including examples for arrays, Lookup Tables, and Time Series Elements) can be found in the GoldSim install directory. In addition to the source files, example solution and project files for Microsoft Visual C++ and Intel Visual Fortran are also included (under General Examples/External).

External Function Calling Sequence

GoldSim makes calls to the DLL external function at different times during the course of a GoldSim simulation:

- Before the simulation starts (while checking model integrity).
- The first time that the External element values are calculated.
- Every time that the input values of the External element change.
- Before each (subsequent) realization.
- After each realization (Cleanup After Realization option only).
- After the simulation finishes.
- If any DLL external function call returns a value of 99.

GoldSim users can control when the DLL is unloaded via the Unload After Each Use and Cleanup After Realization options. These options are selected via checkboxes in the External element properties dialog ("Unload DLL after each use" and "Run Cleanup after each realization"). As the name implies, Unload...
After Each Use will clean up (XFMethod = XF_CLEANUP) and unload the DLL after each calculation call (XFMethod = XFCalculate). Similarly, Cleanup After Realization will clean up and unload the DLL at the end of each realization (if the DLL is loaded).

The DLL external function code can also control when the DLL is unloaded. If any call to a DLL external function returns a status of 99, GoldSim will treat the call as a success, but will clean up and unload the DLL immediately after processing the returned data.

**Before the Simulation**

GoldSim calls the DLL external function before the simulation starts, as part of the process to check the validity of the model. The main reason is to get the number of input and output arguments, to insure that they are consistent with what is specified in the Interface tab. The call sequence is as follows:

1. Load the DLL and check for the existence of the external function
2. Ask for the version number (XFMethod = XF_REP_VERSION).
3. Ask for the number of input and output arguments (XFMethod = XF_REP_ARGUMENTS), and compare to the values determined from the External element interface.
4. Clean up (XFMethod = XF_CLEANUP) and unload the DLL.

If any of these calls fail, or if the number of input or output arguments in step 3 are inconsistent, GoldSim will display an error message and return to the previous state (Edit or Ready mode).

**During Each Realization**

During each realization, GoldSim will call the DLL external function when the corresponding External element needs to be updated. This happens the first time that the Element is referenced, and every subsequent time-step or event update where an input to the Element changes. In this case, the following call sequence is used:

1. Check to see if the DLL is loaded. If so, skip to step 6.
2. Load the DLL and check for the existence of the external function
3. Ask for the version number (XFMethod = XF_REP_VERSION), and write it to the log file.
4. Ask for the number of input and output arguments (XFMethod = XF_REP_ARGUMENTS).
5. Initialize the DLL (XFMethod = XF_INITIALIZE).
6. Calculate (XFMethod = XF_CALCULATE)
7. If Unload After Each Use is specified, clean up (XFMethod = XF_CALCULATE) and unload the DLL.

**Before Each Realization**

Before each realization, GoldSim will check to see if the DLL is loaded for each External element. If the DLL is loaded, GoldSim will reinitialize the DLL (XFMethod = XF_INITIALIZE).

**After Each Realization**

If the Cleanup After Realization option is specified, and the DLL is loaded, GoldSim will clean up (XFMethod = XF_CLEANUP) and unload the DLL after each realization.

**After the Simulation**

After the simulation completes (either successfully or after a fatal error), GoldSim will clean up (XFMethod = XF_CLEANUP) and unload the DLL if it is still loaded.
GoldSim can call DLL external functions by two different mechanisms, depending upon the Separate Process Space option. For each External element, the GoldSim user can select this option in the properties dialog, via the "Run in separate process space" checkbox. If this option is not selected, the DLL will be loaded with the Win32 LoadLibrary function. As a result, this DLL will share the same memory address space as the GoldSim process, and any memory used in the DLL will be charged to the GoldSim process. By default, External elements are created without the Separate Process Space option enabled.

If the Separate Process Space option is selected, GoldSim will call the DLL using a Component Object Model (COM) interface. The interface is implemented as a COM LocalServer executable, which is started when GoldSim first requests to load the DLL. Once the LocalServer is started, it loads the DLL into its own memory address space (separate from GoldSim), and acts as a proxy between GoldSim and the DLL for all external function requests. After the DLL is unloaded, GoldSim frees the COM interface and the LocalServer executable terminates. Because this option loads the DLL into a separate memory space, it may be a better option for DLLs with a large memory footprint.

GoldSim also supports loading of external DLLs that are built as 64-bit libraries. The main advantage for a 64-bit DLL is a significant increase in the amount of virtual memory available (from 4GB to 8TB). Migrating DLL code from 32-bit to 64-bit requires minimal (if any) changes; just install the 64-bit compilers for Visual C++ or Visual Fortran and build with a 64-bit target. No change is GoldSim model configuration is required to use 64-bit DLLs, since GoldSim will automatically determine the type of DLL and call the appropriate interface. However, the following caveats do apply when using 64-bit DLLs in GoldSim:

- 64-bit DLLs can only be run on a computer with a 64-bit Windows operating system. If a DLL needs to run on both 32-bit and 64-bit Windows, it should be a 32-bit DLL (which will run on both 64-bit and 32-bit OS).
- 64-bit DLLs must run in a separate process space.
- For Distributed Processing runs, models that contain 64-bit DLLs can only be launched from a 64-bit Windows operating system. GoldSim will inspect the model to see if it contains a 64-bit DLL, and will disconnect all slaves that are running a 32-bit Windows OS.

To help GoldSim users debug problems with DLL external functions, GoldSim lets users send an error message from the DLL back to GoldSim through the External element interface when the call to an external function fails. The error message is then displayed to the user in a pop-up dialog.

The DLL external function signals the presence of an error message by returning a status value of -2. When GoldSim processes the results of the DLL external function call, it will interpret the first element of the output arguments array (outargs in our source-code example) as a pointer to a memory location where the error string can be found. The memory containing the string must have static scope, so that it will still be available when GoldSim retrieves the string. The string must also be NULL-terminated, even when returning from a FORTRAN DLL. If either of these recommendations are not followed, GoldSim will likely crash when it tries to display the error message.

The following code is an example of a C language function that will properly handle passing a message from a DLL external function to GoldSim. The
ULLONG_PTR is cast to different types on 64-bit (unsigned long) and 32-bit (unsigned __int64), so that it will work for building both 32-bit and 64-bit binaries.

```c
// Utility method used to simplify sending an error message to GoldSim
void CopyMsgToOutputs(const char* sMsg, double* outargs)
{
    // Static character array used to hold the error message.
    // For the current incarnation of GoldSim, this is OK.
    // However, it will not be threadsafe if GoldSim simulations
    // become multithreaded.
    static char sBuffer[81];
    // Clear out any old data from the buffer
    memset(sBuffer, 0, sizeof(sBuffer));

    // Cast the first output array element as a pointer.
    // ULLONG_PTR is used because it will work for both
    // 32-bit and 64-bit DLLs
    ULLONG_PTR* pAddr = (ULLONG_PTR*) outargs;

    // Copy the string data supplied into the static buffer.
    strncpy(sBuffer, sMsg, sizeof(sBuffer) - 1);

    // Copy the static buffer pointer to the first output array
    // element.
    *pAddr = (ULONG_PTR) sBuffer;
}
```

For FORTRAN DLLs, the following code performs the same function:

```fortran
! Utility subroutine to simplify sending an error message to GoldSim
subroutine copy_msg_to_outputs(smsg, outargs)
    implicit none
    character(*) smsg
    real(8) outargs(*)
    ! "Static" character buffer, used to store the error message so
    ! that it can be returned to GoldSim.
    character(80), save :: sbuffer

    ! Create a shared memory variable that can be interpreted either as
    ! integer or real.
    integer(8) ioutput1
    real(8) doutput1
    equivalence(ioutput1, doutput1)

    ! Copy the message into the buffer. Truncate it if it is too long
    ! Make sure that it is null terminated!
    if (len(smsg) .lt. 80) then
        sbuffer = smsg // char(0)
    else
        sbuffer = smsg(1:79) // char(0)
    end if

    ! Since we are sending back to C++, we need an actual address.
    ! The "loc" function is not standard, but it is supported by all
    ! compilers that we checked.
    ioutput1 = loc(sbuffer)
    outargs(1) = doutput1
end subroutine copy_msg_to_outputs
```
Appendix D: GoldSim Units Database

364.4 Smoots plus 1 ear.

Official length of the Harvard Bridge

Appendix Overview

One of the more powerful features of GoldSim is that it is dimensionally-aware. You enable this capability by assigning dimensions to the outputs (and hence to the inputs) of the elements in your model. GoldSim has an extensive internal database of units and conversion factors. This appendix lists all of the units and conversion factors that are built into GoldSim.
Built-in Units and Conversion Factors

All units in GoldSim are defined relative to the following basic units:

- meter (m)
- kilogram (kg)
- second (s)
- Kelvin temperature (K)
- ampere (amp)
- radian (rad)
- Candela (cd)

The following table summarizes all of the built-in units and conversion factors within GoldSim, organized by category:

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Acceleration</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean Acceleration of earth’s gravity</td>
<td>gee</td>
<td>9.80665 m/s²</td>
</tr>
<tr>
<td><strong>Angle</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Degree of Arc</td>
<td>°</td>
<td>0.017453293 rad</td>
</tr>
<tr>
<td>One cycle/revolution</td>
<td>cycle</td>
<td>6.2831853 rad</td>
</tr>
<tr>
<td>Degree of arc</td>
<td>deg</td>
<td>0.017453293 rad</td>
</tr>
<tr>
<td>Minute of Arc</td>
<td>minarc</td>
<td>0.00029088821 rad</td>
</tr>
<tr>
<td>Radian</td>
<td>rad</td>
<td>1 rad</td>
</tr>
<tr>
<td>Revolution (cycle)</td>
<td>rev</td>
<td>6.2831853 rad</td>
</tr>
<tr>
<td>Second of Arc</td>
<td>secarc</td>
<td>4.8481368E-06 rad</td>
</tr>
<tr>
<td><strong>Angular Frequency</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revolutions per minute</td>
<td>pm</td>
<td>6°/s</td>
</tr>
<tr>
<td><strong>Area</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acre</td>
<td>ac</td>
<td>4046.856422 m²</td>
</tr>
<tr>
<td>Acre</td>
<td>acre</td>
<td>4046.856422 m²</td>
</tr>
<tr>
<td>Hectare</td>
<td>ha</td>
<td>10000 m²</td>
</tr>
<tr>
<td><strong>Capacitance</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Farad</td>
<td>Fa</td>
<td>1 s4-amp2/kg-m²</td>
</tr>
<tr>
<td><strong>Charge</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coulomb of Charge</td>
<td>Co</td>
<td>1 s-amp</td>
</tr>
<tr>
<td>GigaCoulomb of Charge</td>
<td>GCo</td>
<td>1000000000 s-amp</td>
</tr>
<tr>
<td>KiloCoulomb of charge</td>
<td>kCo</td>
<td>1000 s-amp</td>
</tr>
<tr>
<td>MillaCoulomb of charge</td>
<td>mCo</td>
<td>0.001 s-amp</td>
</tr>
<tr>
<td>MegaCoulomb of charge</td>
<td>MCo</td>
<td>1000000 s-amp</td>
</tr>
<tr>
<td>NanoCoulomb of charge</td>
<td>nCo</td>
<td>1.00E-09 s-amp</td>
</tr>
<tr>
<td>PicoCoulomb of charge</td>
<td>pCo</td>
<td>1.00E-12 s-amp</td>
</tr>
<tr>
<td>TeraCoulomb of charge</td>
<td>TCo</td>
<td>1E+12 s-amp</td>
</tr>
<tr>
<td>Unit</td>
<td>Abbreviation</td>
<td>Definition</td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>MicroCoulomb of charge</td>
<td>uCo</td>
<td>1.00E-06 s-amp</td>
</tr>
<tr>
<td><strong>Currency</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>US Dollar</td>
<td>$</td>
<td>(user defined)</td>
</tr>
<tr>
<td>Euro</td>
<td>EUR</td>
<td>(user defined)</td>
</tr>
<tr>
<td>British Pound</td>
<td>GBP</td>
<td>(user defined)</td>
</tr>
<tr>
<td>Japanese Yen</td>
<td>YEN</td>
<td>(user defined)</td>
</tr>
<tr>
<td>Australian Dollar</td>
<td>AUD</td>
<td>(user defined)</td>
</tr>
<tr>
<td>Brazilian Real</td>
<td>BRL</td>
<td>(user defined)</td>
</tr>
<tr>
<td>Canadian Dollar</td>
<td>CAD</td>
<td>(user defined)</td>
</tr>
<tr>
<td>Chinese Yuan</td>
<td>CNY</td>
<td>(user defined)</td>
</tr>
<tr>
<td>Czech Koruna</td>
<td>CZK</td>
<td>(user defined)</td>
</tr>
<tr>
<td>Danish Krona</td>
<td>DKK</td>
<td>(user defined)</td>
</tr>
<tr>
<td>Hong Kong Dollar</td>
<td>HKD</td>
<td>(user defined)</td>
</tr>
<tr>
<td>Hungarian Forint</td>
<td>HUF</td>
<td>(user defined)</td>
</tr>
<tr>
<td>Mexican Peso</td>
<td>MXN</td>
<td>(user defined)</td>
</tr>
<tr>
<td>New Zealand Dollar</td>
<td>NZD</td>
<td>(user defined)</td>
</tr>
<tr>
<td>Norwegian Krone</td>
<td>NOK</td>
<td>(user defined)</td>
</tr>
<tr>
<td>Russian Rouble</td>
<td>RUB</td>
<td>(user defined)</td>
</tr>
<tr>
<td>Singapore Dollar</td>
<td>SGD</td>
<td>(user defined)</td>
</tr>
<tr>
<td>Swedish Krona</td>
<td>SEK</td>
<td>(user defined)</td>
</tr>
<tr>
<td>Swiss Franc</td>
<td>CHF</td>
<td>(user defined)</td>
</tr>
<tr>
<td>South African Rand</td>
<td>ZAR</td>
<td>(user defined)</td>
</tr>
<tr>
<td>Thousand US Dollar</td>
<td>k$</td>
<td>1000 $</td>
</tr>
<tr>
<td>Thousand Euro</td>
<td>kEUR</td>
<td>1000 EUR</td>
</tr>
<tr>
<td>Thousand British Pound</td>
<td>kGBP</td>
<td>1000 GBP</td>
</tr>
<tr>
<td>Thousand Japanese Yen</td>
<td>kYEN</td>
<td>1000 YEN</td>
</tr>
<tr>
<td>Thousand Australian Dollar</td>
<td>kAUD</td>
<td>1000 AUD</td>
</tr>
<tr>
<td>Thousand Brazilian Real</td>
<td>kBRL</td>
<td>1000 BRL</td>
</tr>
<tr>
<td>Thousand Canadian Dollar</td>
<td>kCAD</td>
<td>1000 CAD</td>
</tr>
<tr>
<td>Thousand Chinese Yuan</td>
<td>kCNY</td>
<td>1000 CNY</td>
</tr>
<tr>
<td>Thousand Czech Koruna</td>
<td>kCZK</td>
<td>1000 CZK</td>
</tr>
<tr>
<td>Thousand Danish Krone</td>
<td>kDKK</td>
<td>1000 DKK</td>
</tr>
<tr>
<td>Thousand Hong Kong Dollar</td>
<td>kHKD</td>
<td>1000 HKD</td>
</tr>
<tr>
<td>Thousand Hungarian Forint</td>
<td>kHUF</td>
<td>1000 HUF</td>
</tr>
<tr>
<td>Thousand Mexican Peso</td>
<td>kMXN</td>
<td>1000 MZN</td>
</tr>
<tr>
<td>Thousand New Zealand Dollar</td>
<td>kNZD</td>
<td>1000 NZD</td>
</tr>
<tr>
<td>Thousand Norwegian Krone</td>
<td>kNOK</td>
<td>1000 NOK</td>
</tr>
<tr>
<td>Thousand Russian Rouble</td>
<td>kRUB</td>
<td>1000 RUB</td>
</tr>
<tr>
<td>Thousand Singapore Dollar</td>
<td>kSGD</td>
<td>1000 SGD</td>
</tr>
<tr>
<td>Thousand Swedish Krona</td>
<td>kSEK</td>
<td>1000 SEK</td>
</tr>
<tr>
<td>Thousand Swiss Franc</td>
<td>kCHF</td>
<td>1000 CHF</td>
</tr>
<tr>
<td><strong>Unit</strong></td>
<td><strong>Abbreviation</strong></td>
<td><strong>Definition</strong></td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>Thousand South African Rand</td>
<td>kZAR</td>
<td>1000 ZAR</td>
</tr>
<tr>
<td>Million US Dollar</td>
<td>$</td>
<td>1000000 $</td>
</tr>
<tr>
<td>Million Euro</td>
<td>MEUR</td>
<td>1000000 EUR</td>
</tr>
<tr>
<td>Million British Pound</td>
<td>MGBP</td>
<td>1000000 GBP</td>
</tr>
<tr>
<td>Million Japanese Yen</td>
<td>MYEN</td>
<td>1000000 YEN</td>
</tr>
<tr>
<td>Million Australian Dollar</td>
<td>MAUD</td>
<td>1000000 AUD</td>
</tr>
<tr>
<td>Million Brazilian Real</td>
<td>MBRL</td>
<td>1000000 BRL</td>
</tr>
<tr>
<td>Million Canadian Dollar</td>
<td>MCAD</td>
<td>1000000 CAD</td>
</tr>
<tr>
<td>Million Chinese Yuan</td>
<td>MCNY</td>
<td>1000000 CNY</td>
</tr>
<tr>
<td>Million Czech Koruna</td>
<td>MCZK</td>
<td>1000000 CZK</td>
</tr>
<tr>
<td>Million Danish Krone</td>
<td>MDKK</td>
<td>1000000 DKK</td>
</tr>
<tr>
<td>Million Hong Kong Dollar</td>
<td>MHKD</td>
<td>1000000 HKD</td>
</tr>
<tr>
<td>Million Hungarian Forint</td>
<td>MHUF</td>
<td>1000000 HUF</td>
</tr>
<tr>
<td>Million Mexican Peso</td>
<td>MMXN</td>
<td>1000000 MZN</td>
</tr>
<tr>
<td>Million New Zealand Dollar</td>
<td>MNZD</td>
<td>1000000 NZD</td>
</tr>
<tr>
<td>Million Norwegian Krone</td>
<td>MNOK</td>
<td>1000000 NOK</td>
</tr>
<tr>
<td>Million Russian Rouble</td>
<td>MRUB</td>
<td>1000000 RUB</td>
</tr>
<tr>
<td>Million Singapore Dollar</td>
<td>MSGD</td>
<td>1000000 SGD</td>
</tr>
<tr>
<td>Million Swedish Krona</td>
<td>MSEK</td>
<td>1000000 SEK</td>
</tr>
<tr>
<td>Million Swiss Franc</td>
<td>MCHF</td>
<td>1000000 CHF</td>
</tr>
<tr>
<td>Million South African Rand</td>
<td>MZAR</td>
<td>1000000 ZAR</td>
</tr>
<tr>
<td><strong>Current</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ampere</td>
<td>amp</td>
<td>1 amp</td>
</tr>
<tr>
<td>GigaAmpere</td>
<td>Gamp</td>
<td>1000000000 amp</td>
</tr>
<tr>
<td>KiloAmpere</td>
<td>kamp</td>
<td>1000 amp</td>
</tr>
<tr>
<td>MilliAmpere</td>
<td>mamp</td>
<td>0.001 amp</td>
</tr>
<tr>
<td>MegaAmpere</td>
<td>Mamp</td>
<td>1000000 amp</td>
</tr>
<tr>
<td>NanoAmpere</td>
<td>namp</td>
<td>1.00E-09 amp</td>
</tr>
<tr>
<td>PicoAmpere</td>
<td>pamp</td>
<td>1.00E-12 amp</td>
</tr>
<tr>
<td>TeraAmpere</td>
<td>Tamp</td>
<td>1E+12 amp</td>
</tr>
<tr>
<td>MicroAmpere</td>
<td>uamp</td>
<td>1.00E-06 amp</td>
</tr>
<tr>
<td><strong>Dose</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GigaGray (dose absorbed)</td>
<td>GGy</td>
<td>1000000000 m2/s2</td>
</tr>
<tr>
<td>GigaSievert (dose equivalent)</td>
<td>GSV</td>
<td>1000000000 m2/s2</td>
</tr>
<tr>
<td>Gray (dose absorbed)</td>
<td>Gy</td>
<td>1 m2/s2</td>
</tr>
<tr>
<td>KiloGray (dose absorbed)</td>
<td>kGy</td>
<td>1000 m2/s2</td>
</tr>
<tr>
<td>KiloSievert (dose equivalent)</td>
<td>kSV</td>
<td>1000 m2/s2</td>
</tr>
<tr>
<td>MilliGray (dose absorbed)</td>
<td>mGy</td>
<td>0.001 m2/s2</td>
</tr>
<tr>
<td>MegaGray (dose absorbed)</td>
<td>MGy</td>
<td>1000000 m2/s2</td>
</tr>
<tr>
<td>MilliSievert (dose equivalent)</td>
<td>mSv</td>
<td>0.001 m2/s2</td>
</tr>
<tr>
<td>MegaSievert (dose equivalent)</td>
<td>MSv</td>
<td>1000000 m2/s2</td>
</tr>
<tr>
<td>Unit</td>
<td>Abbreviation</td>
<td>Definition</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------</td>
<td>------------------</td>
</tr>
<tr>
<td>NanoGray (dose absorbed)</td>
<td>nGy</td>
<td>1.00E-09 m²/s²</td>
</tr>
<tr>
<td>NanoSievert (dose equivalent)</td>
<td>nSv</td>
<td>1.00E-09 m²/s²</td>
</tr>
<tr>
<td>PicoGray (dose absorbed)</td>
<td>pGy</td>
<td>1.00E-12 m²/s²</td>
</tr>
<tr>
<td>PicoSievert (dose equivalent)</td>
<td>pSv</td>
<td>1.00E-12 m²/s²</td>
</tr>
<tr>
<td>RAD dose</td>
<td>RADD</td>
<td>0.01 m²/s²</td>
</tr>
<tr>
<td>REM dose</td>
<td>REM</td>
<td>0.01 m²/s²</td>
</tr>
<tr>
<td>MilliREM dose</td>
<td>mREM</td>
<td>1.00E-05 m²/s²</td>
</tr>
<tr>
<td>Sievert (dose equivalent)</td>
<td>Sv</td>
<td>1 m²/s²</td>
</tr>
<tr>
<td>TeraGray (dose absorbed)</td>
<td>TGy</td>
<td>1E+12 m²/s²</td>
</tr>
<tr>
<td>TeraSievert (dose equivalent)</td>
<td>TSv</td>
<td>1E+12 m²/s²</td>
</tr>
<tr>
<td>MicroGray (dose absorbed)</td>
<td>uGy</td>
<td>1.00E-06 m²/s²</td>
</tr>
<tr>
<td>MicroSievert (dose equivalent)</td>
<td>uSv</td>
<td>1.00E-06 m²/s²</td>
</tr>
</tbody>
</table>

**Electrical Resistance**

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>GigaOhm</td>
<td>Gohm</td>
<td>10000000000 kg-m²/s²-3-amp²</td>
</tr>
<tr>
<td>KiloOhm</td>
<td>kohm</td>
<td>1000 kg-m²/s²-3-amp²</td>
</tr>
<tr>
<td>MilliOhm</td>
<td>mohm</td>
<td>0.001 kg-m²/s²-3-amp²</td>
</tr>
<tr>
<td>MegaOhm</td>
<td>Mohm</td>
<td>1000000 kg-m²/s²-3-amp²</td>
</tr>
<tr>
<td>PicoOhm</td>
<td>pohm</td>
<td>1.00E-12 kg-m²/s²-3-amp²</td>
</tr>
<tr>
<td>NanoOhm</td>
<td>nohm</td>
<td>1.00E-09 kg-m²/s²-3-amp²</td>
</tr>
<tr>
<td>Ohm</td>
<td>ohm</td>
<td>1 kg-m²/s²</td>
</tr>
<tr>
<td>TeraOhm</td>
<td>Tohm</td>
<td>1E+12 kg-m²/s²-3-amp²</td>
</tr>
<tr>
<td>MicroOhm</td>
<td>uohm</td>
<td>1.00E-06 kg-m²/s²-3-amp²</td>
</tr>
</tbody>
</table>

**Energy**

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>British Thermal Unit (Int’l)</td>
<td>BTU</td>
<td>1055.056 kg-m²/s²</td>
</tr>
<tr>
<td>Calorie</td>
<td>cal</td>
<td>4.1868 kg-m²/s²</td>
</tr>
<tr>
<td>Electron Volt</td>
<td>eV</td>
<td>1.60E-19 kg-m²/s²</td>
</tr>
<tr>
<td>GigaCalorie</td>
<td>Gcal</td>
<td>4186800000 kg-m²/s²</td>
</tr>
<tr>
<td>GigaElectron volt</td>
<td>GeV</td>
<td>1.60E-10 kg-m²/s²</td>
</tr>
<tr>
<td>GigaJoule</td>
<td>GJ</td>
<td>1000000000 kg-m²/s²</td>
</tr>
<tr>
<td>Joule</td>
<td>J</td>
<td>1 kg-m²/s²</td>
</tr>
<tr>
<td>KiloCalorie</td>
<td>kcal</td>
<td>4186.8 kg-m²/s²</td>
</tr>
<tr>
<td>KiloElectron volt</td>
<td>keV</td>
<td>1.60E-16 kg-m²/s²</td>
</tr>
<tr>
<td>KiloJoule</td>
<td>kJ</td>
<td>1000 kg-m²/s²</td>
</tr>
<tr>
<td>Kilowatt-hour</td>
<td>kwh</td>
<td>3600000 kg-m²/s²</td>
</tr>
<tr>
<td>MilliCalorie</td>
<td>mcal</td>
<td>0.0041868 kg-m²/s²</td>
</tr>
<tr>
<td>MegaCalorie</td>
<td>Mcal</td>
<td>4186800 kg-m²/s²</td>
</tr>
<tr>
<td>MilliElectron volt</td>
<td>meV</td>
<td>1.60E-22 kg-m²/s²</td>
</tr>
<tr>
<td>MegaElectron volt</td>
<td>MeV</td>
<td>1.60E-13 kg-m²/s²</td>
</tr>
<tr>
<td>MilliJoule</td>
<td>mJ</td>
<td>0.001 kg-m²/s²</td>
</tr>
<tr>
<td>MegaJoule</td>
<td>MJ</td>
<td>1000000 kg-m²/s²</td>
</tr>
<tr>
<td>Unit</td>
<td>Abbreviation</td>
<td>Definition</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>NanoCalorie</td>
<td>ncal</td>
<td>4.19E-09 kg-m2/s2</td>
</tr>
<tr>
<td>NanoElectron volt</td>
<td>neV</td>
<td>1.60E-28 kg-m2/s2</td>
</tr>
<tr>
<td>NanoJoule</td>
<td>nJ</td>
<td>1.00E-09 kg-m2/s2</td>
</tr>
<tr>
<td>PicoCalorie</td>
<td>pcal</td>
<td>4.19E-12 kg-m2/s2</td>
</tr>
<tr>
<td>PicoElectron volt</td>
<td>peV</td>
<td>1.60E-31 kg-m2/s2</td>
</tr>
<tr>
<td>PicoJoule</td>
<td>pJ</td>
<td>1.00E-12 kg-m2/s2</td>
</tr>
<tr>
<td>TeraCalorie</td>
<td>Tcal</td>
<td>4.1868E+12 kg-m2/s2</td>
</tr>
<tr>
<td>TeraElectron volt</td>
<td>TeV</td>
<td>1.60E-07 kg-m2/s2</td>
</tr>
<tr>
<td>TeraJoule</td>
<td>TJ</td>
<td>1E+12 kg-m2/s2</td>
</tr>
<tr>
<td>MicroCalorie</td>
<td>ucal</td>
<td>4.19E-06 kg-m2/s2</td>
</tr>
<tr>
<td>MicroElectron volt</td>
<td>ueV</td>
<td>1.60E-25 kg-m2/s2</td>
</tr>
<tr>
<td>MicroJoule</td>
<td>uJ</td>
<td>1.00E-06 kg-m2/s2</td>
</tr>
</tbody>
</table>

### Flux (Volume)

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acre-feet/day</td>
<td>afd</td>
<td>0.01427641 m3/s</td>
</tr>
<tr>
<td>US Barrels/day</td>
<td>bpd</td>
<td>1.84E-06 m3/s</td>
</tr>
<tr>
<td>Cubic feet per second</td>
<td>cfs</td>
<td>0.028316847 m3/s</td>
</tr>
<tr>
<td>US Gallons per minute</td>
<td>gpm</td>
<td>6.31E-05 m3/s</td>
</tr>
<tr>
<td>Million gallons per day</td>
<td>MGD</td>
<td>0.043812639 m3/s</td>
</tr>
</tbody>
</table>

### Force

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dyne</td>
<td>dyne</td>
<td>1.00E-05 kg-m/s2</td>
</tr>
<tr>
<td>GramForce</td>
<td>gf</td>
<td>0.00980665 kg-m/s2</td>
</tr>
<tr>
<td>GigaNewton</td>
<td>GN</td>
<td>1000000000 kg-m/s2</td>
</tr>
<tr>
<td>Kilogram Force</td>
<td>kgf</td>
<td>9.80665 kg-m/s2</td>
</tr>
<tr>
<td>1,000 Pound force</td>
<td>kip</td>
<td>4448.221909 kg-m/s2</td>
</tr>
<tr>
<td>KiloNewton</td>
<td>kN</td>
<td>1000 kg-m/s2</td>
</tr>
<tr>
<td>Pound force</td>
<td>lbf</td>
<td>4.448221909 kg-m/s2</td>
</tr>
<tr>
<td>Milligram force</td>
<td>mgf</td>
<td>9.81E-06 kg-m/s2</td>
</tr>
<tr>
<td>MilliNewton</td>
<td>mN</td>
<td>0.001 kg-m/s2</td>
</tr>
<tr>
<td>MegaNewton</td>
<td>MN</td>
<td>1000000 kg-m/s2</td>
</tr>
<tr>
<td>Newton</td>
<td>N</td>
<td>1 kg-m/s2</td>
</tr>
<tr>
<td>NanoNewton</td>
<td>nN</td>
<td>1.00E-09 kg-m/s2</td>
</tr>
<tr>
<td>Ounce force</td>
<td>ozf</td>
<td>0.278013869 kg-m/s2</td>
</tr>
<tr>
<td>PicoNewton</td>
<td>pN</td>
<td>1.00E-12 kg-m/s2</td>
</tr>
<tr>
<td>TeraNewton</td>
<td>TN</td>
<td>1E+12 kg-m/s2</td>
</tr>
<tr>
<td>Ton force</td>
<td>tonf</td>
<td>8896.443819 kg-m/s2</td>
</tr>
<tr>
<td>MicroNewton</td>
<td>uN</td>
<td>1.00E-06 kg-m/s2</td>
</tr>
</tbody>
</table>

### Frequency (non-angular, Rate)

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>PicoHertz (frequency)</td>
<td>pHz</td>
<td>1.00E-12 1/s</td>
</tr>
<tr>
<td>Becquerel</td>
<td>Bq</td>
<td>1 1/s</td>
</tr>
<tr>
<td>Curie</td>
<td>Ci</td>
<td>37000000000 1/s</td>
</tr>
<tr>
<td>GigaBecquerel</td>
<td>GBq</td>
<td>1000000000 1/s</td>
</tr>
<tr>
<td>Unit</td>
<td>Abbreviation</td>
<td>Definition</td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>GigaHertz (frequency)</td>
<td>GHz</td>
<td>1000000000 1/s</td>
</tr>
<tr>
<td>Hertz (frequency)</td>
<td>Hz</td>
<td>1 1/s</td>
</tr>
<tr>
<td>KiloBecquerel</td>
<td>kBq</td>
<td>1000 1/s</td>
</tr>
<tr>
<td>KiloHertz (frequency)</td>
<td>kHz</td>
<td>1000 1/s</td>
</tr>
<tr>
<td>MilliBecquerel</td>
<td>mBq</td>
<td>0.001 1/s</td>
</tr>
<tr>
<td>MegaBecquerel</td>
<td>MBq</td>
<td>1000000 1/s</td>
</tr>
<tr>
<td>MilliHertz (frequency)</td>
<td>mHz</td>
<td>0.001 1/s</td>
</tr>
<tr>
<td>MegaHertz (frequency)</td>
<td>MHz</td>
<td>1000000 1/s</td>
</tr>
<tr>
<td>NanoBecquerel</td>
<td>nBq</td>
<td>1.00E-09 1/s</td>
</tr>
<tr>
<td>NanoHertz (frequency)</td>
<td>nHz</td>
<td>1.00E-09 1/s</td>
</tr>
<tr>
<td>PicoBecquerel</td>
<td>pBq</td>
<td>1.00E-12 1/s</td>
</tr>
<tr>
<td>Picocurie</td>
<td>pCi</td>
<td>0.037 1/s</td>
</tr>
<tr>
<td>TeraBecquerel</td>
<td>TBq</td>
<td>1E+12 1/s</td>
</tr>
<tr>
<td>TeraHertz (frequency)</td>
<td>THz</td>
<td>1E+12 1/s</td>
</tr>
<tr>
<td>MicroBecquerel</td>
<td>uBq</td>
<td>1.00E-06 1/s</td>
</tr>
<tr>
<td>MicroCurie</td>
<td>uCi</td>
<td>37000 1/s</td>
</tr>
<tr>
<td>MicroHertz (frequency)</td>
<td>uHz</td>
<td>1.00E-06 1/s</td>
</tr>
<tr>
<td><strong>Illuminance</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lambert</td>
<td>lamb</td>
<td>10000 cd/m2</td>
</tr>
<tr>
<td>Lux (1 lm/m²)</td>
<td>lx</td>
<td>1 cd/m2</td>
</tr>
<tr>
<td><strong>Inverse Area</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Miles per gallon</td>
<td>mpg</td>
<td>425143.68321/m²</td>
</tr>
<tr>
<td><strong>Items</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item</td>
<td>item</td>
<td>(dimensionless)</td>
</tr>
<tr>
<td>Persons</td>
<td>pers</td>
<td>1 item</td>
</tr>
<tr>
<td>Thousand Persons</td>
<td>kpers</td>
<td>1000 item</td>
</tr>
<tr>
<td>Million Persons</td>
<td>Mpers</td>
<td>1000000 item</td>
</tr>
<tr>
<td><strong>Length</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Angstrom</td>
<td>Ang</td>
<td>1.00E-10 m</td>
</tr>
<tr>
<td>Centimeter</td>
<td>cm</td>
<td>0.01 m</td>
</tr>
<tr>
<td>Fathom</td>
<td>fath</td>
<td>1.8288 m</td>
</tr>
<tr>
<td>Furlong</td>
<td>flng</td>
<td>201.168 m</td>
</tr>
<tr>
<td>Foot (US)</td>
<td>ft, '</td>
<td>0.3048 m</td>
</tr>
<tr>
<td>GigaMeter</td>
<td>Gm</td>
<td>1000000000 m</td>
</tr>
<tr>
<td>Inch</td>
<td>in, &quot;</td>
<td>0.0254 m</td>
</tr>
<tr>
<td>KiloMeter</td>
<td>km</td>
<td>1000 m</td>
</tr>
<tr>
<td>Light Year</td>
<td>ly</td>
<td>9.46E+15 m</td>
</tr>
<tr>
<td>Meter</td>
<td>m</td>
<td>1 m</td>
</tr>
<tr>
<td>Mile</td>
<td>mi</td>
<td>1609.344 m</td>
</tr>
<tr>
<td>Mil=0.001 inch</td>
<td>mil</td>
<td>2.54E-05 m</td>
</tr>
<tr>
<td>MilliMeter</td>
<td>mm</td>
<td>0.001 m</td>
</tr>
</tbody>
</table>
## Built-in Units and Conversion Factors

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>MegaMeter</td>
<td>Mm</td>
<td>1000000 m</td>
</tr>
<tr>
<td>Nautical Mile</td>
<td>naut</td>
<td>1852 m</td>
</tr>
<tr>
<td>NanoMeter</td>
<td>nm</td>
<td>1.00E-09 m</td>
</tr>
<tr>
<td>PicoMeter</td>
<td>pm</td>
<td>1.00E-12 m</td>
</tr>
<tr>
<td>Rod</td>
<td>rd</td>
<td>5.0292 m</td>
</tr>
<tr>
<td>TeraMeter</td>
<td>Tm</td>
<td>1E+12 m</td>
</tr>
<tr>
<td>Yard</td>
<td>yard</td>
<td>0.9144 m</td>
</tr>
</tbody>
</table>

### Luminous Intensity

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Candela</td>
<td>cd</td>
<td>1 cd</td>
</tr>
<tr>
<td>GigaCandela</td>
<td>Gcd</td>
<td>1000000000 cd</td>
</tr>
<tr>
<td>KiloCandela</td>
<td>kcd</td>
<td>1000 cd</td>
</tr>
<tr>
<td>Lumen (cd/Sterradian)</td>
<td>lm</td>
<td>0.079577472 cd</td>
</tr>
<tr>
<td>MilliCandela</td>
<td>mcd</td>
<td>0.001 cd</td>
</tr>
<tr>
<td>NanoCandela</td>
<td>ncd</td>
<td>1.00E-09 cd</td>
</tr>
<tr>
<td>PicoCandela</td>
<td>pcd</td>
<td>1.00E-12 cd</td>
</tr>
<tr>
<td>TeraCandela</td>
<td>Tcd</td>
<td>1E+12 cd</td>
</tr>
<tr>
<td>MicroCandela</td>
<td>ucd</td>
<td>1.00E-06 cd</td>
</tr>
<tr>
<td>MegaCandela</td>
<td>Mcd</td>
<td>10000000 cd</td>
</tr>
</tbody>
</table>

### Mass

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gram</td>
<td>g</td>
<td>0.001 kg</td>
</tr>
<tr>
<td>GigaGram</td>
<td>Gg</td>
<td>1000000 kg</td>
</tr>
<tr>
<td>KiloGram</td>
<td>kg</td>
<td>1 kg</td>
</tr>
<tr>
<td>Pound (mass)</td>
<td>lbm</td>
<td>0.4535924 kg</td>
</tr>
<tr>
<td>MilliGram</td>
<td>mg</td>
<td>1.00E-06 kg</td>
</tr>
<tr>
<td>MegaGram</td>
<td>Mg</td>
<td>1000 kg</td>
</tr>
<tr>
<td>NanoGram</td>
<td>ng</td>
<td>1.00E-12 kg</td>
</tr>
<tr>
<td>Ounce (mass)</td>
<td>ozm</td>
<td>0.028349525 kg</td>
</tr>
<tr>
<td>PicoGram</td>
<td>pg</td>
<td>1.00E-15 kg</td>
</tr>
<tr>
<td>Slug Mass</td>
<td>slug</td>
<td>14.5939039 kg</td>
</tr>
<tr>
<td>TeraGram</td>
<td>Tg</td>
<td>1000000000 kg</td>
</tr>
<tr>
<td>Ton (mass)</td>
<td>tonm</td>
<td>907.1848 kg</td>
</tr>
<tr>
<td>Tonne</td>
<td>tonne</td>
<td>1000 kg</td>
</tr>
<tr>
<td>MicroGram</td>
<td>ug</td>
<td>1.00E-09 kg</td>
</tr>
</tbody>
</table>

### Math Constants

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parts per billion</td>
<td>ppb</td>
<td>1.00E-09</td>
</tr>
<tr>
<td>Parts per million</td>
<td>ppm</td>
<td>1.00E-06</td>
</tr>
<tr>
<td>Percentage</td>
<td>%</td>
<td>0.01</td>
</tr>
</tbody>
</table>

### Permeability (seepage)

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Darcy</td>
<td>Darcy</td>
<td>9.87E-13 m2</td>
</tr>
<tr>
<td>Millidarcy</td>
<td>md</td>
<td>9.87E-16 m2</td>
</tr>
<tr>
<td>Unit</td>
<td>Abbreviation</td>
<td>Definition</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td><strong>Power</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GigaWatt</td>
<td>GW</td>
<td>$1000000000$ kg-m2/s3</td>
</tr>
<tr>
<td>Horsepower (550 ft-lb/sec)</td>
<td>hp</td>
<td>$745.6999209$ kg-m2/s3</td>
</tr>
<tr>
<td>KiloWatt</td>
<td>kW</td>
<td>$1000$ kg-m2/s3</td>
</tr>
<tr>
<td>MilliWatt</td>
<td>mW</td>
<td>$0.001$ kg-m2/s3</td>
</tr>
<tr>
<td>MegaWatt</td>
<td>MW</td>
<td>$1000000$ kg-m2/s3</td>
</tr>
<tr>
<td>NanoWatt</td>
<td>nW</td>
<td>$1.00E-09$ kg-m2/s3</td>
</tr>
<tr>
<td>PicoWatt</td>
<td>pW</td>
<td>$1.00E-12$ kg-m2/s3</td>
</tr>
<tr>
<td>TeraWatt</td>
<td>TW</td>
<td>$1E+12$ kg-m2/s3</td>
</tr>
<tr>
<td>MicroWatt</td>
<td>uW</td>
<td>$1.00E-06$ kg-m2/s3</td>
</tr>
<tr>
<td>Watt</td>
<td>W</td>
<td>$1$ kg-m2/s3</td>
</tr>
<tr>
<td><strong>Pressure, Stress</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Atmosphere</td>
<td>atm</td>
<td>$101325$ kg/m-s2</td>
</tr>
<tr>
<td>Bar</td>
<td>bar</td>
<td>$100000$ kg/m-s2</td>
</tr>
<tr>
<td>GigaBar</td>
<td>Gbar</td>
<td>$1E+14$ kg/m-s2</td>
</tr>
<tr>
<td>GigaPascal</td>
<td>GPa</td>
<td>$1000000000$ kg/m-s2</td>
</tr>
<tr>
<td>KiloBar</td>
<td>kbar</td>
<td>$100000000$ kg/m-s2</td>
</tr>
<tr>
<td>KiloPond</td>
<td>kp</td>
<td>$98066.5$ kg-m/s2</td>
</tr>
<tr>
<td>KiloPascal</td>
<td>kPa</td>
<td>$1000$ kg/m-s2</td>
</tr>
<tr>
<td>MilliBar</td>
<td>mbar</td>
<td>$100$ kg/m-s2</td>
</tr>
<tr>
<td>MegaBar</td>
<td>Mbar</td>
<td>$1E+11$ kg/m-s2</td>
</tr>
<tr>
<td>MilliPascal</td>
<td>mPa</td>
<td>$0.001$ kg/m-s2</td>
</tr>
<tr>
<td>MegaPascal</td>
<td>MPa</td>
<td>$1000000$ kg/m-s2</td>
</tr>
<tr>
<td>NanoBar</td>
<td>nbar</td>
<td>$0.0001$ kg/m-s2</td>
</tr>
<tr>
<td>NanoPascal</td>
<td>nPa</td>
<td>$1.00E-09$ kg/m-s2</td>
</tr>
<tr>
<td>Pascal</td>
<td>Pa</td>
<td>$1$ kg/m-s2</td>
</tr>
<tr>
<td>PicoBar</td>
<td>pbar</td>
<td>$1.00E-07$ kg/m-s2</td>
</tr>
<tr>
<td>PicoPascal</td>
<td>pPa</td>
<td>$1.00E-12$ kg/m-s2</td>
</tr>
<tr>
<td>Pound per square foot</td>
<td>psf</td>
<td>$47.88026215$ kg/m-s2</td>
</tr>
<tr>
<td>Pound per square inch</td>
<td>psi</td>
<td>$6894.757749$ kg/m-s2</td>
</tr>
<tr>
<td>TeraBar</td>
<td>Tbar</td>
<td>$1.00E+17$ kg/m-s2</td>
</tr>
<tr>
<td>Torr (mm Hg)</td>
<td>torr</td>
<td>$133.3221913$ kg/m-s2</td>
</tr>
<tr>
<td>TeraPascal</td>
<td>TPa</td>
<td>$1E+12$ kg/m-s2</td>
</tr>
<tr>
<td>MicroBar</td>
<td>ubar</td>
<td>$0.1$ kg/m-s2</td>
</tr>
<tr>
<td>MicroPascal</td>
<td>uPa</td>
<td>$1.00E-06$ kg/m-s2</td>
</tr>
<tr>
<td><strong>Quantity of Matter</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GigaMol</td>
<td>Gmol</td>
<td>$1000000000$ mol</td>
</tr>
<tr>
<td>KiloMole</td>
<td>kmol</td>
<td>$1000$ mol</td>
</tr>
<tr>
<td>MilliMole</td>
<td>mmol</td>
<td>$0.001$ mol</td>
</tr>
<tr>
<td>MegaMole</td>
<td>Mmol</td>
<td>$1000000$ mol</td>
</tr>
<tr>
<td>Mole</td>
<td>mol</td>
<td>$1$ mol</td>
</tr>
</tbody>
</table>
### Built-in Units and Conversion Factors

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NanoMole</td>
<td>nmol</td>
<td>1.00E-09 mol</td>
</tr>
<tr>
<td>PicoMole</td>
<td>pmol</td>
<td>1.00E-12 mol</td>
</tr>
<tr>
<td>TeraMole</td>
<td>Tmol</td>
<td>1E+12 mol</td>
</tr>
<tr>
<td>MicroMole</td>
<td>umol</td>
<td>1.00E-06 mol</td>
</tr>
</tbody>
</table>

#### Temperature

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Celsius temperature</td>
<td>C</td>
<td>1 K</td>
</tr>
<tr>
<td>Celsius degree</td>
<td>Cdeg</td>
<td>1 K</td>
</tr>
<tr>
<td>Fahrenheit temperature</td>
<td>F</td>
<td>0.555555556 K</td>
</tr>
<tr>
<td>Fahrenheit Degree</td>
<td>Fdeg</td>
<td>0.555555556 K</td>
</tr>
<tr>
<td>GigaKelvin temperature</td>
<td>GK</td>
<td>1000000000 K</td>
</tr>
<tr>
<td>Kelvin temperature</td>
<td>K</td>
<td>1 K</td>
</tr>
<tr>
<td>KiloKelvin temperature</td>
<td>kK</td>
<td>1000 K</td>
</tr>
<tr>
<td>MilliKelvin temperature</td>
<td>mK</td>
<td>0.001 K</td>
</tr>
<tr>
<td>MegaKelvin temperature</td>
<td>MK</td>
<td>1000000 K</td>
</tr>
<tr>
<td>NanoKalvin temperature</td>
<td>nK</td>
<td>1.00E-09 K</td>
</tr>
<tr>
<td>PicoKelvin temperature</td>
<td>pK</td>
<td>1.00E-12 K</td>
</tr>
<tr>
<td>Rankine temperature</td>
<td>R</td>
<td>0.555555556 K</td>
</tr>
<tr>
<td>TerraKelvin temperature</td>
<td>TK</td>
<td>1E+12 K</td>
</tr>
<tr>
<td>MicroKelvin temperature</td>
<td>uK</td>
<td>1.00E-06 K</td>
</tr>
</tbody>
</table>

#### Time

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>a, yr</td>
<td>31557600 s</td>
</tr>
<tr>
<td>Day</td>
<td>d, day</td>
<td>86400 s</td>
</tr>
<tr>
<td>GigaSeconds of time</td>
<td>Gs</td>
<td>1000000000 s</td>
</tr>
<tr>
<td>Hour</td>
<td>hr</td>
<td>3600 s</td>
</tr>
<tr>
<td>Kilosecond of time</td>
<td>ks</td>
<td>1000 s</td>
</tr>
<tr>
<td>Minute of time</td>
<td>min</td>
<td>60 s</td>
</tr>
<tr>
<td>Month</td>
<td>mon</td>
<td>2629800 s</td>
</tr>
<tr>
<td>MilliSecond of time</td>
<td>ms</td>
<td>0.001 s</td>
</tr>
<tr>
<td>MegaSecond of time</td>
<td>Ms</td>
<td>1000000 s</td>
</tr>
<tr>
<td>NanoSecond of time</td>
<td>ns</td>
<td>1.00E-09 s</td>
</tr>
<tr>
<td>PicoSecond of time</td>
<td>ps</td>
<td>1.00E-12 s</td>
</tr>
<tr>
<td>Second of time</td>
<td>s, sec</td>
<td>1 s</td>
</tr>
<tr>
<td>TerraSecond of time</td>
<td>Ts</td>
<td>1E+12 s</td>
</tr>
<tr>
<td>MicroSecond of time</td>
<td>us</td>
<td>1.00E-06 s</td>
</tr>
<tr>
<td>Week</td>
<td>week</td>
<td>604800 s</td>
</tr>
<tr>
<td>Date</td>
<td>date</td>
<td>86400 s</td>
</tr>
<tr>
<td>Date and time</td>
<td>datetime</td>
<td>86400 s</td>
</tr>
</tbody>
</table>

#### Velocity

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feet per minute</td>
<td>fpm</td>
<td>0.00508 m/s</td>
</tr>
<tr>
<td>Feet per second</td>
<td>fps</td>
<td>0.3048 m/s</td>
</tr>
<tr>
<td>Kilometer per hour</td>
<td>kph</td>
<td>0.277777778 m/s</td>
</tr>
<tr>
<td>Unit</td>
<td>Abbreviation</td>
<td>Definition</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>Knots</td>
<td>kt</td>
<td>0.514444444 m/s</td>
</tr>
<tr>
<td>Miles per hour</td>
<td>mph</td>
<td>0.44704 m/s</td>
</tr>
<tr>
<td>Viscosity (Absolute)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Centipose</td>
<td>cp</td>
<td>0.001 kg/m/s</td>
</tr>
<tr>
<td>Poise</td>
<td>poise</td>
<td>0.1 kg/m/s</td>
</tr>
<tr>
<td>Viscosity (Kinematic)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stoke</td>
<td>stoke</td>
<td>0.0001 m2/s</td>
</tr>
<tr>
<td>Voltage</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GigaVolt</td>
<td>GV</td>
<td>1000000000 kg-m2/s3-amp</td>
</tr>
<tr>
<td>KiloVolt</td>
<td>kV</td>
<td>1000 kg-m2/s3-amp</td>
</tr>
<tr>
<td>MilliVolt</td>
<td>mV</td>
<td>0.001 kg-m2/s3-amp</td>
</tr>
<tr>
<td>MegaVolt</td>
<td>MV</td>
<td>1000000 kg-m2/s3-amp</td>
</tr>
<tr>
<td>NanoVolt</td>
<td>nV</td>
<td>1.00E-9 kg-m2/s3-amp</td>
</tr>
<tr>
<td>PicoVolt</td>
<td>pV</td>
<td>1.00E-12 kg-m2/s3-amp</td>
</tr>
<tr>
<td>TeraVolt</td>
<td>TV</td>
<td>1E+12 kg-m2/s3-amp</td>
</tr>
<tr>
<td>MicroVolt</td>
<td>uV</td>
<td>1.00E-06 kg-m2/s3-amp</td>
</tr>
<tr>
<td>Volt</td>
<td>V</td>
<td>1 kg-m2/s3-amp</td>
</tr>
<tr>
<td>Volume</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acre-feet</td>
<td>af</td>
<td>1233.48183754752 m3</td>
</tr>
<tr>
<td>Thousand Acre-feet</td>
<td>kaf</td>
<td>1233481.83754752 m3</td>
</tr>
<tr>
<td>Million Acre-feet</td>
<td>Maf</td>
<td>1233481837.54752 m3</td>
</tr>
<tr>
<td>Barrel (US, oil)</td>
<td>bbl</td>
<td>0.1589873 m3</td>
</tr>
<tr>
<td>Barrel (US, dry)</td>
<td>bblDry</td>
<td>0.11563 m3</td>
</tr>
<tr>
<td>Barrel (US, liquid)</td>
<td>bblLiq</td>
<td>0.11924 m3</td>
</tr>
<tr>
<td>Bushel</td>
<td>bushel</td>
<td>0.03523907 m3</td>
</tr>
<tr>
<td>Cubic Centimeter</td>
<td>cc</td>
<td>1.00E-06 m3</td>
</tr>
<tr>
<td>Cup, US</td>
<td>cup</td>
<td>0.000236588 m3</td>
</tr>
<tr>
<td>Fluid ounce</td>
<td>floz</td>
<td>2.96E-05 m3</td>
</tr>
<tr>
<td>Gallon (US)</td>
<td>gal</td>
<td>0.003785412 m3</td>
</tr>
<tr>
<td>Gallon (Imperial)</td>
<td>gali</td>
<td>0.00454609 m3</td>
</tr>
<tr>
<td>Gallon (US)</td>
<td>galus</td>
<td>0.003785412 m3</td>
</tr>
<tr>
<td>GigaLitre</td>
<td>Gl, GL</td>
<td>1000000 m3</td>
</tr>
<tr>
<td>KiloLitre</td>
<td>kl, kL</td>
<td>1 m3</td>
</tr>
<tr>
<td>Litre</td>
<td>l, L</td>
<td>0.001 m3</td>
</tr>
<tr>
<td>MilliLitre</td>
<td>ml, mL</td>
<td>1.00E-06 m3</td>
</tr>
<tr>
<td>MegaLitre</td>
<td>ML, ML</td>
<td>1000 m3</td>
</tr>
<tr>
<td>NanoLitre</td>
<td>nl</td>
<td>1.00E-12 m3</td>
</tr>
<tr>
<td>Pint, US liquid</td>
<td>pint</td>
<td>0.000473177 m3</td>
</tr>
<tr>
<td>PicoLitre</td>
<td>pL, pL</td>
<td>1.00E-15 m3</td>
</tr>
<tr>
<td>Quart (US)</td>
<td>qt</td>
<td>0.000946353 m3</td>
</tr>
<tr>
<td>Unit</td>
<td>Abbreviation</td>
<td>Definition</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Standard cubic foot</td>
<td>stcf</td>
<td>0.028316847 m³</td>
</tr>
<tr>
<td>Tablespoon</td>
<td>tbsp</td>
<td>1.48E-05 m³</td>
</tr>
<tr>
<td>TeraLitre</td>
<td>Tl, TL</td>
<td>1000000000 m³</td>
</tr>
<tr>
<td>Teaspoon</td>
<td>tsp</td>
<td>4.93E-06 m³</td>
</tr>
<tr>
<td>MicroLitre</td>
<td>ul, uL</td>
<td>1.00E-09 m³</td>
</tr>
</tbody>
</table>
Appendix E: Database Input File Formats

Art and science cannot exist but in minutely organized particulars.

William Blake, To the Public

Appendix Overview

In simulations which require a great deal of input, it may be desirable that the simulation model can access the various data sources directly to ensure the quality of the data transfer.

To facilitate this, GoldSim data entry elements can be linked directly to an ODBC-compliant database.

Read more: Linking Elements to a Database (page 1031).

After defining the linkage, you can then instruct GoldSim to download the data at any time. When it does this, GoldSim internally records the time and date at which the download occurred, along with other reference information retrieved from the database (e.g., document references), and this is stored with the model in the Run Log. This allows you to confirm that the correct data were loaded into your model, and provides very strong and defensible quality control over your model input data.

GoldSim can import from three different types of databases: a Generic Database, a Simple GoldSim Database, and an Extended GoldSim Database. This appendix describes the details of the structure and format for each of these three database types.

In this Appendix

This appendix discusses the following:

- Creating a Generic Database
- Creating a Simple GoldSim Database
- Creating an Extended GoldSim Database
Creating a Generic Database

The generic database format requirements are very general:

- The database must be ODBC compliant;
- The selected table must have a field (column) which contains unique IDs which will be used to map data to specific GoldSim elements. These IDs would typically be the GoldSim element names (but they do not have to be);
- The table must have one or more columns containing the data items to be downloaded.

Note that by assigning multiple records and using multiple fields for each ID, you can download vector and matrix data from the generic database.

Read more: Downloading from a Generic Database (page 1035).

The file GenericDatabase.gsm in the General Examples/Database folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) provides an example of how to use a Generic database. This folder also includes the sample database referenced by this file (GenericDatabase.accdb), created using Microsoft Access. In order to use the GoldSim file, you will need to add the database as data sources to your computer.

Read more: Adding Data Sources to Your Computer (page 1033).

Creating a Simple GoldSim Database

A Simple GoldSim database contains the following three tables:

- tbl_Parameter
- tbl_Parameter_Reference
- tbl_Probability_Value_Pairs

Each of these tables is discussed in detail below.

Parameter Table

The Parameter Table (tbl_Parameter) must contain the following fields (which must be named as shown below):

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID</td>
<td>AutoNumber</td>
<td>Unique integer number assigned to each element.</td>
</tr>
<tr>
<td>Parameter_Name</td>
<td>Text</td>
<td>The ID of the GoldSim element. Note that the length of an element ID in GoldSim is limited to 30 characters.</td>
</tr>
<tr>
<td>Parameter_Path</td>
<td>Memo</td>
<td>The full path to the element in GoldSim. This attribute is optional. GoldSim tries to find a record set in the database using the name and path information. If this query is not successful it will try again just querying the name. If a path is specified it must end with ''.</td>
</tr>
<tr>
<td>Type_Code</td>
<td>Text</td>
<td>Code to describe the parameter type (see below).</td>
</tr>
</tbody>
</table>
Field Name | Type | Description
--- | --- | ---
ModDate | Date/Time | Last Date that the parameter properties were changed. Note that this field is for information purposes only and is not used by GoldSim.
Current | Yes/No | This version of the parameter (this record) is considered active in GoldSim if this flag is set to Yes. Note that only one parameter with the same name (and path) should have the current flag set to true.
Description | Text | Description of the parameter, inserted into the Description field for the element.
Unit | Text | Unit abbreviation for the parameter. See Appendix D for unit abbreviations. The unit should be specified without parentheses or brackets.
Arg_1 | Number (double) | Value for the first argument for the parameter. All parameters have at least one argument.
Arg_2 | Number (double) | Value for the second argument for the parameter.
Arg_3 | Number (double) | Value for the third argument for the parameter.
Arg_4 | Number (double) | Value for the fourth argument for the parameter.

**Note:** Particular care must be taken when importing Stochastics from a database, as different Stochastics could have different dimensions (e.g., a Binomial is always dimensionless; a Boolean is always a condition). If a Stochastic is defined in your model as a particular type of distribution whose output is inconsistent with that of the distribution that you are trying to import from the database, GoldSim will display an error.

**Note:** Care must also be taken when importing Stochastics that represent temperatures. This is because temperatures have an absolute reference (i.e., absolute zero). *Differences* between temperatures are expressed in ‘deg’ units (Cdeg, Fdeg). This can lead to errors when importing Stochastics, since GoldSim assumes a single unit for the distribution parameters, while some types of distributions would require two different types of units. For example, a Normal distribution representing a temperature would require the Mean to be specified in absolute units (e.g., C) and the Standard Deviation to be specified in difference units (e.g., Cdeg). If faced with this problem, there are several approaches for addressing this: 1) Use a distribution type where all parameters have the same units (e.g., triangular, uniform, cumulative); or 2) Specify and import the distributions as dimensionless and then apply a unit to the sampled value using an Expression element.
Warning: The special GoldSim units “date” and “datetime” cannot be used when importing from a database.

Note that the Parameter_Path does not need to be defined (it can be blank). In this case, it does not matter where the element exists in the model. If you specify a path, it must start and end with a backslash (e.g., `\container1\container2\`). If you want to specify the top-level Container (the model root), you should use a single backslash.

If you specify a path, GoldSim first tries to find a record with the specified element name and path. If it does not find it, it tries again, ignoring the path.

If GoldSim finds multiple records with the same name and path, and both have the Current field marked “Yes”, it will issue an error message (i.e., if multiple records in the database have the same name and path, only one record can have the Current flag set to Yes).

The codes for the various parameter types, and their required arguments are shown below:

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Type_Code</th>
<th>Arg_1</th>
<th>Arg_2</th>
<th>Arg_3</th>
<th>Arg_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>constant (Data element)</td>
<td>100</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant (vector Data element)</td>
<td>101</td>
<td>Number of rows</td>
<td>1 (Number of columns)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant (matrix Data element)</td>
<td>102</td>
<td>Number of rows</td>
<td>Number of columns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>uniform</td>
<td>2100</td>
<td>Minimum</td>
<td>Maximum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>log-uniform</td>
<td>2101</td>
<td>Minimum</td>
<td>Maximum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>normal</td>
<td>2200</td>
<td>Mean</td>
<td>Std. Deviation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>truncated normal</td>
<td>2202</td>
<td>Mean</td>
<td>Std. Deviation</td>
<td>Minimum</td>
<td>Maximum</td>
</tr>
<tr>
<td>log-normal (geometric input)</td>
<td>2300</td>
<td>Geometric Mean</td>
<td>Geometric Std. Deviation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>truncated log-normal (geometric input)</td>
<td>2302</td>
<td>Geometric Mean</td>
<td>Geometric Std. Deviation</td>
<td>Minimum</td>
<td>Maximum</td>
</tr>
<tr>
<td>log-normal (true mean input)</td>
<td>2330</td>
<td>True Mean</td>
<td>True Std. Deviation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>truncated log-normal (true mean input)</td>
<td>2332</td>
<td>True Mean</td>
<td>True Std. Deviation</td>
<td>Minimum</td>
<td>Maximum</td>
</tr>
<tr>
<td>triangular</td>
<td>2400</td>
<td>Minimum</td>
<td>Most Likely</td>
<td>Maximum</td>
<td></td>
</tr>
<tr>
<td>log triangular</td>
<td>2401</td>
<td>Minimum</td>
<td>Most Likely</td>
<td>Maximum</td>
<td></td>
</tr>
<tr>
<td>Distribution</td>
<td>Type Code</td>
<td>Arg_1</td>
<td>Arg_2</td>
<td>Arg_3</td>
<td>Arg_4</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------</td>
<td>----------------</td>
<td>------------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>triangular (10/90)</td>
<td>2402</td>
<td>10th percentile</td>
<td>Most Likely</td>
<td>90th percentile</td>
<td></td>
</tr>
<tr>
<td>log triangular (10/90)</td>
<td>2403</td>
<td>10th percentile</td>
<td>Most Likely</td>
<td>90th percentile</td>
<td></td>
</tr>
<tr>
<td>cumulative</td>
<td>2500</td>
<td></td>
<td>references Probability_Value_Pairs table (see below)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Log-cumulative</td>
<td>2501</td>
<td></td>
<td>references Probability_Value_Pairs table (see below)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>discrete</td>
<td>2600</td>
<td></td>
<td>references Probability_Value_Pairs table (see below)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>poisson</td>
<td>2700</td>
<td></td>
<td>Expected Value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>beta (generalized)</td>
<td>2800</td>
<td>Mean</td>
<td>Std. Deviation</td>
<td>Minimum</td>
<td>Maximum</td>
</tr>
<tr>
<td>beta (success, failure)</td>
<td>2804</td>
<td>Sucesses</td>
<td>Failures</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BetaPERT</td>
<td>4200</td>
<td>Minimum</td>
<td>Most Likely</td>
<td>Maximum</td>
<td></td>
</tr>
<tr>
<td>BetarPERT 10/90</td>
<td>4201</td>
<td>10th percentile</td>
<td>Most Likely</td>
<td>90th percentile</td>
<td></td>
</tr>
<tr>
<td>gamma</td>
<td>2900</td>
<td>Mean</td>
<td>Std. Deviation</td>
<td>Minimum</td>
<td>Maximum</td>
</tr>
<tr>
<td>truncated gamma</td>
<td>2902</td>
<td>Mean</td>
<td>Std. Deviation</td>
<td>Minimum</td>
<td>Maximum</td>
</tr>
<tr>
<td>weibull</td>
<td>3000</td>
<td>Minimum</td>
<td>Weibull Slope</td>
<td>Mean-Minimum</td>
<td></td>
</tr>
<tr>
<td>truncated weibull</td>
<td>3002</td>
<td>Minimum</td>
<td>Weibull Slope</td>
<td>Mean-Minimum</td>
<td>Maximum</td>
</tr>
<tr>
<td>binomial</td>
<td>3100</td>
<td># of Picks (Batch size)</td>
<td>Probability of Success</td>
<td></td>
<td></td>
</tr>
<tr>
<td>boolean</td>
<td>3200</td>
<td>Probability of True</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Student’s t</td>
<td>3300</td>
<td>Degrees of freedom</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exponential</td>
<td>3400</td>
<td>Mean</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pareto</td>
<td>3500</td>
<td>a</td>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>truncated Pareto</td>
<td>3502</td>
<td>a</td>
<td>b</td>
<td>Maximum</td>
<td></td>
</tr>
<tr>
<td>Distribution</td>
<td>Type_Code</td>
<td>Arg_1</td>
<td>Arg_2</td>
<td>Arg_3</td>
<td>Arg_4</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------</td>
<td>------------------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>negative binomial</td>
<td>3600</td>
<td>Successes</td>
<td>Probability of Success</td>
<td></td>
<td></td>
</tr>
<tr>
<td>extreme value (minimum)</td>
<td>3800</td>
<td>Location</td>
<td>Scale</td>
<td></td>
<td></td>
</tr>
<tr>
<td>extreme value (maximum)</td>
<td>3803</td>
<td>Location</td>
<td>Scale</td>
<td></td>
<td></td>
</tr>
<tr>
<td>extreme probability (minimum)</td>
<td>3900</td>
<td>Number of Samples</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>extreme probability (maximum)</td>
<td>3903</td>
<td>Number of Samples</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pearson type III</td>
<td>4000</td>
<td>Location</td>
<td>Scale</td>
<td>Shape</td>
<td></td>
</tr>
<tr>
<td>sampled results (no extrapolation)</td>
<td>4100</td>
<td>references Probability_Value_Pairs table (see below)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sampled results (extrapolation)</td>
<td>4103</td>
<td>references Probability_Value_Pairs table (see below)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that for the Discrete, Cumulative and Sampled Results distributions, the first argument references the `val_pair_UID` field in the Probability Value Pairs table (described below).

The Parameter Reference Table (tbl_Parameter_Reference) allows you to specify reference information for the element.

The table has the following fields:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter_UID</td>
<td>Text</td>
<td>Primary Key – link from UID in Parameter Table</td>
</tr>
<tr>
<td>GS_Parameter_Note</td>
<td>Text</td>
<td>The text in this field overwrites the Note associated with the element in GoldSim. If left blank here, the Note in GoldSim is not overwritten.</td>
</tr>
</tbody>
</table>

When GoldSim imports text from a database into a GoldSim Note, it automatically converts text to hyperlinks in the Note under the following circumstances:

- Any text beginning the prefixes listed below is converted to a hyperlink. The hyperlink terminates when a space in encountered. As a result, hyperlinks with spaces will not be recognized by GoldSim.
  - http://
  - www.
Creating a Simple GoldSim Database

- ftp://
- ftp.
- file://

- If the character @ is encountered, and text before and after the @ not delimited by a space or a line break will be considered part of the hyperlink.

**Array Values Table**

The Array Values Table (tbl_Array_Values) is used to store an arbitrary number of array values for a vector or matrix Data element.

The table has the following fields:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Number</td>
<td>Unique integer number assigned to each row in the table.</td>
</tr>
<tr>
<td>Array_UID</td>
<td>Number</td>
<td>The parameter ID from the Parameter Table (tbl_Parameter) identifying the Data element.</td>
</tr>
<tr>
<td>Value</td>
<td>Number</td>
<td>The value for the specified row and column of the array.</td>
</tr>
<tr>
<td>Row</td>
<td>Number</td>
<td>The row of the array. This index is 1-based.</td>
</tr>
<tr>
<td>Column</td>
<td>Number</td>
<td>The column of the array. This index is 1-based. It must be set to 1 for vectors.</td>
</tr>
</tbody>
</table>

**Probability Value Pairs Table**

The Probability Value Pair Table (tbl_Probability_Value_Pairs) is used to store an arbitrary number of value pairs used in defining discrete, cumulative and sampled results distributions. No other type of element uses this table.

The table has the following fields:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Val_pair_UID</td>
<td>Number</td>
<td>Identifies a set of value-probability-pairs that are used by discrete, cumulative and sampled results distributions. This distribution must specify this ID in arg_1 in tbl_Parameter.</td>
</tr>
<tr>
<td>Probability</td>
<td>Number</td>
<td>The probability (for discrete distributions) or cumulative probability (for Cumulative distributions) of the data pair (dimensionless). Ignored for sampled results distributions.</td>
</tr>
<tr>
<td>Value</td>
<td>Number</td>
<td>The value corresponding with the defined probability level for Discrete, Cumulative and Sampled Results distributions. Uses the unit defined in the parameter table.</td>
</tr>
</tbody>
</table>

**Example File and Database Template**

The file SimpleDatabase.gsm in the General Examples/Database folder of your GoldSim directory (accessed by selecting File | Open Example... from the main menu) provides an example of how to use a Simple GoldSim database. This folder also includes the sample database referenced by this file (SimpleDatabase.accdb), created using Microsoft Access. In order to use the
Creating an Extended GoldSim Database

When creating a GoldSim file, you will need to add the database as data sources to your computer.

Read more: Adding Data Sources to Your Computer (page 1033).

The Database subfolder also includes a template for creating Simple GoldSim databases (SimpleDatabaseTemplate.accdb). This template file includes two additional tables (providing parameter type codes and unit abbreviations) which can be used to support the implementation of custom forms which allow the user to pick a parameter type code and unit from a list.

Creating an Extended GoldSim Database

An Extended GoldSim database must contain the following three tables:

- GS_Parameter
- GS_Parameter_Value
- GS_Value_Component

Each table is described in detail below.

Note: The tables described below can contain additional fields not used by GoldSim. When importing information, GoldSim will ignore any extra fields.

Parameter Table

The Parameter Table (GS_Parameter) has one record for each linked Element. It contains basic descriptive information about the Element, and an index (Parameter_ID) which links it into the GS_Parameter_Value table. It must contain the following fields:

<table>
<thead>
<tr>
<th>No.</th>
<th>Field</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Parameter_ID</td>
<td>Unique integer number assigned to each element</td>
</tr>
<tr>
<td>2</td>
<td>Parameter_Name</td>
<td>Key field which must match the GoldSim element ID</td>
</tr>
<tr>
<td>3</td>
<td>Description</td>
<td>Text description of the element</td>
</tr>
<tr>
<td>4</td>
<td>Units</td>
<td>String with GoldSim abbreviations for units of the data (see Appendix D for correct unit abbreviations)</td>
</tr>
</tbody>
</table>
| 5   | Parameter_Code    | 100  Data element  
1nn  Array Data element, where nn is the # of columns (01 for vectors)  
2100  Stochastic: uniform  
2101  Stochastic: log-uniform  
2200  Stochastic: normal  
2202  Stochastic: truncated normal  
2300  Stochastic: lognormal (geometric mean)  
2302  Stochastic: truncated lognormal (geometric mean)  
2330  Stochastic: lognormal (true mean) |
<table>
<thead>
<tr>
<th>No.</th>
<th>Field</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2332</td>
<td>Stochastic: truncated lognormal (true mean)</td>
<td></td>
</tr>
<tr>
<td>2400</td>
<td>Stochastic: triangular</td>
<td></td>
</tr>
<tr>
<td>2401</td>
<td>Stochastic: log-triangular</td>
<td></td>
</tr>
<tr>
<td>2402</td>
<td>Stochastic: triangular (10/90)</td>
<td></td>
</tr>
<tr>
<td>2403</td>
<td>Stochastic: log-triangular (10/90)</td>
<td></td>
</tr>
<tr>
<td>2500</td>
<td>Stochastic: cumulative</td>
<td></td>
</tr>
<tr>
<td>2501</td>
<td>Stochastic: Log-cumulative</td>
<td></td>
</tr>
<tr>
<td>2600</td>
<td>Stochastic: discrete:</td>
<td></td>
</tr>
<tr>
<td>2700</td>
<td>Stochastic: Poisson</td>
<td></td>
</tr>
<tr>
<td>2800</td>
<td>Stochastic: Generalized Beta</td>
<td></td>
</tr>
<tr>
<td>2804</td>
<td>Stochastic: Beta (Success, Failure)</td>
<td></td>
</tr>
<tr>
<td>2900</td>
<td>Stochastic: Gamma</td>
<td></td>
</tr>
<tr>
<td>2902</td>
<td>Stochastic: Truncated Gamma</td>
<td></td>
</tr>
<tr>
<td>3000</td>
<td>Stochastic: Weibull</td>
<td></td>
</tr>
<tr>
<td>3002</td>
<td>Stochastic: Truncated Weibull</td>
<td></td>
</tr>
<tr>
<td>3100</td>
<td>Stochastic: Binomial</td>
<td></td>
</tr>
<tr>
<td>3200</td>
<td>Stochastic: Boolean</td>
<td></td>
</tr>
<tr>
<td>3300</td>
<td>Stochastic: Student’s t</td>
<td></td>
</tr>
<tr>
<td>3400</td>
<td>Stochastic: Exponential</td>
<td></td>
</tr>
<tr>
<td>3500</td>
<td>Stochastic: Pareto</td>
<td></td>
</tr>
<tr>
<td>3502</td>
<td>Stochastic: Truncated Pareto</td>
<td></td>
</tr>
<tr>
<td>3600</td>
<td>Stochastic: Negative Binomial</td>
<td></td>
</tr>
<tr>
<td>3800</td>
<td>Stochastic: extreme value (minimum)</td>
<td></td>
</tr>
<tr>
<td>3803</td>
<td>Stochastic: extreme value (maximum)</td>
<td></td>
</tr>
<tr>
<td>3900</td>
<td>Stochastic: extreme probability (minimum)</td>
<td></td>
</tr>
<tr>
<td>3903</td>
<td>Stochastic: extreme probability (maximum)</td>
<td></td>
</tr>
<tr>
<td>4000</td>
<td>Stochastic: Pearson type III</td>
<td></td>
</tr>
<tr>
<td>4100</td>
<td>Stochastic: sampled results</td>
<td></td>
</tr>
<tr>
<td>4103</td>
<td>Stochastic: sampled results (extrapolation)</td>
<td></td>
</tr>
<tr>
<td>4200</td>
<td>Stochastic: BetaPERT</td>
<td></td>
</tr>
<tr>
<td>4201</td>
<td>Stochastic: BetaPERT (10/90)</td>
<td></td>
</tr>
<tr>
<td>5100</td>
<td>1-D Table</td>
<td></td>
</tr>
<tr>
<td>52nn</td>
<td>2-D Table, where nn is the no. of columns</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>indep_row_units</td>
<td>String with GoldSim abbreviations for units of a table element’s Row independent variable (only required for tables)</td>
</tr>
<tr>
<td>7</td>
<td>indep_col_units</td>
<td>String with GoldSim abbreviations for units of the 2-D table element’s Column independent variable (only required for 2-D tables)</td>
</tr>
</tbody>
</table>
Creating an Extended GoldSim Database

### Parameter Value Table

The Parameter Value Table (GS_Parameter_Value) has one record for each Element and each effective date. Each record must have a unique Effective_Date and Value_ID. The Value_ID is an index which is used to link into the actual data values, which are stored in table GS_Value_Component. The GS_Parameter_Value table must contain the following fields:

<table>
<thead>
<tr>
<th>No.</th>
<th>Field</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Parameter_ID</td>
<td>Key to link from items in table GS_Parameter</td>
</tr>
<tr>
<td>2</td>
<td>Value_ID</td>
<td>Unique integer key for value record(s) in table GS_Value_Component</td>
</tr>
<tr>
<td>3</td>
<td>Effective_Date</td>
<td>The effective date for this item, in format YYYY-MM-DD HH:MM:SS</td>
</tr>
<tr>
<td>4</td>
<td>Reference_Document</td>
<td>Written by GoldSim to the Run Log (an optional string)</td>
</tr>
<tr>
<td>5</td>
<td>Document_ID</td>
<td>Written by GoldSim to the Run Log (an optional string)</td>
</tr>
</tbody>
</table>

---

**Note:** Particular care must be taken when importing Stochastics from a database, as different Stochastics could have different dimensions (e.g., a Binomial is always dimensionless; a Boolean is always a condition). If a Stochastic is defined in your model as a particular type of distribution whose output is inconsistent with that of the distribution that you are trying to import from the database, GoldSim will display an error.

**Note:** Care must also be taken when importing Stochastics that represent temperatures. This is because temperatures have an absolute reference (i.e., absolute zero). Differences between temperatures are expressed in ‘deg’ units (Cdeg, Fdeg). This can lead to errors when importing Stochastics, since GoldSim assumes a single unit for the distribution parameters, while some types of distributions would require two different types of units. For example, a Normal distribution representing a temperature would require the Mean to be specified in absolute units (e.g., C) and the Standard Deviation to be specified in difference units (e.g., Cdeg). If faced with this problem, there are several approaches for addressing this: 1) Use a distribution type where all parameters have the same units (e.g., triangular, uniform, cumulative); or 2) Specify and import the distributions as dimensionless and then apply a unit to the sampled value using an Expression element.

**Read more:** [Dealing with Temperature Units](#) (page 92).

**Warning:** The special GoldSim units “date” and “datetime” cannot be used when importing from a database.
Creating an Extended GoldSim Database

<table>
<thead>
<tr>
<th>No.</th>
<th>Field</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>DTN</td>
<td>Written by GoldSim to the Run Log (an optional string)</td>
</tr>
<tr>
<td>7</td>
<td>MOL</td>
<td>Written by GoldSim to the Run Log (an optional string)</td>
</tr>
<tr>
<td>8</td>
<td>DOC_Path</td>
<td>Network path for the source document (required only for File elements, as discussed below)</td>
</tr>
<tr>
<td>9</td>
<td>DOC_SIG</td>
<td>CRC signature for File source document (a string required only for File elements)</td>
</tr>
<tr>
<td>10</td>
<td>Parameter_Note</td>
<td>Text that is imported into the element’s Note. This must be a “Memo” field, and does not support rich text or HTML. Only plain text can be imported.</td>
</tr>
</tbody>
</table>

The CRC signature is an alphanumeric code that can be used to uniquely identify whether the file contents have changed. When you download a file, GoldSim compares the CRC signature of the downloaded file with the original signature that was stored in the database. If these are not identical (indicating that the file has been changed), the download will fail.

You can generate a CRC signature for a file using the EFIViewer, a small utility program that is installed with GoldSim.

The Value Component Table (GS_Value_Component) stores the actual data values. There are one or more records for each data value. Each record must contain the following fields:

<table>
<thead>
<tr>
<th>No.</th>
<th>Field</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Value_ID</td>
<td>The index used to link from the GS_Parameter_Value table</td>
</tr>
<tr>
<td>2</td>
<td>Component_ID</td>
<td>Unique index</td>
</tr>
<tr>
<td>3</td>
<td>Type_Code</td>
<td>Row number for sorting rows in vectors and matrices (used only arrays).</td>
</tr>
<tr>
<td>4 - 63</td>
<td>Value_Column_1, …, Value_Column_60</td>
<td>Data values, to support tables and matrices with up to 60 columns.</td>
</tr>
</tbody>
</table>

For a Data element, the data value is stored in Value_Column_1.

For vector Data elements, the data for each item is stored in Value_Column_1, and there should be one record for each row (item) in the vector. For matrix Data elements, the data for each row of the matrix is stored in Value_Column_1 through Value_Column_nn (where nn is as specified in the Parameter_Code field of the Parameter Table), and there should be one record for each row of the matrix.

For a 1-D Table element, the independent variable values are stored in Value_Column_1, and the dependent variable values are stored in Value_Column_2. There is one record for each row in the table.

For a 2-D Table element, the row independent variable values are stored in Value_Column_1, and the dependent variable values are stored in Value_Column_2 through Value_Column_n, where n is one greater than the number of columns in the table. The first record for a 2-D Table element contains values for the column independent variable in Value_Column_2.
through Value_Column_n, and the successive records contain values for the Row independent variable followed by values for the dependent variable.

**Note:** Matrices and 2-D tables can have no more than 60 columns.

For Stochastic elements other than discrete, cumulative and sampled results, the arguments are stored, in sequence, in the Value_Column_1 … Value_Column_n entries. The order of the arguments for each type of Stochastic is listed below:

<table>
<thead>
<tr>
<th>Stochastic</th>
<th>Order of Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>minimum, maximum</td>
</tr>
<tr>
<td>Log-uniform</td>
<td>minimum, maximum</td>
</tr>
<tr>
<td>Normal</td>
<td>mean, standard deviation</td>
</tr>
<tr>
<td>Truncated Normal</td>
<td>mean, standard deviation, minimum, maximum</td>
</tr>
<tr>
<td>Log-normal (geometric)</td>
<td>geometric mean, geometric standard deviation</td>
</tr>
<tr>
<td>Truncated Log-normal (geometric)</td>
<td>geometric mean, geometric standard deviation, minimum, maximum</td>
</tr>
<tr>
<td>Log-normal (true mean)</td>
<td>true mean, true standard deviation</td>
</tr>
<tr>
<td>Truncated Log-normal (true mean)</td>
<td>true mean, true standard deviation, minimum, maximum</td>
</tr>
<tr>
<td>Triangular</td>
<td>Minimum (or 10%), most likely, maximum (or 90%)</td>
</tr>
<tr>
<td>Log-triangular</td>
<td>Minimum (or 10%), most likely, maximum (or 90%)</td>
</tr>
<tr>
<td>Poisson</td>
<td>expected value</td>
</tr>
<tr>
<td>Beta</td>
<td>Successes, Failures</td>
</tr>
<tr>
<td>Generalized Beta</td>
<td>Mean, standard deviation, minimum, maximum</td>
</tr>
<tr>
<td>BetaPERT</td>
<td>Minimum (or 10%), most likely, maximum (or 90%)</td>
</tr>
<tr>
<td>Gamma</td>
<td>mean, standard deviation</td>
</tr>
<tr>
<td>Truncated Gamma</td>
<td>mean, standard deviation, minimum, maximum</td>
</tr>
<tr>
<td>Weibull</td>
<td>minimum, slope, mean-minimum</td>
</tr>
<tr>
<td>Truncated Weibull</td>
<td>minimum, slope, mean-minimum</td>
</tr>
<tr>
<td>Binomial</td>
<td># picks, probability of success</td>
</tr>
<tr>
<td>Boolean</td>
<td>probability of true</td>
</tr>
<tr>
<td>Student’s t</td>
<td>degrees of freedom</td>
</tr>
<tr>
<td>Exponential</td>
<td>mean</td>
</tr>
<tr>
<td>Pareto</td>
<td>a, b</td>
</tr>
<tr>
<td>Truncated Pareto</td>
<td>a, b, maximum</td>
</tr>
<tr>
<td>Negative Binomial</td>
<td>successes, probability of success</td>
</tr>
<tr>
<td>Extreme Value (minimum)</td>
<td>location, scale</td>
</tr>
<tr>
<td>Extreme Value (maximum)</td>
<td>location, scale</td>
</tr>
<tr>
<td>Extreme Probability (minimum)</td>
<td>number of samples</td>
</tr>
<tr>
<td>Extreme Probability (maximum)</td>
<td>number of samples</td>
</tr>
</tbody>
</table>
For a discrete, cumulative or sampled results Stochastic element type, there are multiple rows in the table, with one row for each value. Value_Column_1 contains the probability values, and Value_Column_2 contains the result values. For a sampled results distribution, there is only one value (the result) and this is placed in Value_Column_2 (Value_Column_1 is ignored).

For a Sampled Results distribution, there are multiple rows in the table, with one row for each value. Value_Column_2 contains the result values. Any probabilities entered in Value_Column_1 are ignored (all results have equal weights).

The file ExtendedDatabase.gsm in the General Examples/Database folder of your GoldSim directory (accessed by selecting **File | Open Example...** from the main menu) provides an example of how to use an Extended GoldSim database. This folder also includes the sample database referenced by this file (ExtendedDatabase.accdb), created using Microsoft Access. In order to use the GoldSim file, you will need to add the database as data sources to your computer.

*Read more:* [Adding Data Sources to Your Computer](#) (page 1033).
Appendix F: Integration Methods and Timestepping Algorithm

On two occasions I have been asked [by members of Parliament], 'Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?' I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question.

Charles Babbage

Appendix Overview

The elements and links in a GoldSim model represent a system of equations. Except in the simplest cases, these are systems of differential equations, and they are often nonlinear and discontinuous. In general, the systems of equations that GoldSim must solve will not have an analytical solution (i.e., they cannot be solved exactly), and must be solved numerically (i.e., using an algorithm that provides a numerical approximation to the actual solution).

In order to effectively use GoldSim, particularly for complex problems, it is important to have a basic understanding of the factors affecting the accuracy of your model, and the nature of the numerical approximations used by GoldSim.

In this Appendix

This appendix provides a brief discussion of these numerical algorithms.

This appendix discusses the following:

- Factors Affecting the Accuracy of Simulation Models
- Primary Numerical Approximations in GoldSim
- Summary of GoldSim’s Dynamic Timestepping Algorithm
Factors Affecting the Accuracy of Simulation Models

A simulation model is an abstract representation of an actual (or hypothetical) system. By definition, it is a simplification of reality, with the goals being to include those aspects that are assumed to be important and omit those which are considered to be nonessential, so as to derive useful predictions of system performance in an efficient manner.

In addition, for most real world systems, there will be significant uncertainty regarding the processes that are controlling the system and the parameter values describing those processes. As a result, the most important factor impacting the accuracy of your model is the degree to which your conceptual and mathematical model have captured reality and the degree to which you have quantitatively represented your uncertainty in the system. In most real world systems that you would simulate in GoldSim, the uncertainty in the simulated result due to your uncertainty in the processes and parameters controlling the system will be far greater than any inaccuracies introduced by the numerical solution method. As a result, it will generally be more worthwhile for you to spend your time ensuring that your model captures the key aspects of the system realistically rather that worrying about the numerical accuracy of the solution.

Having said that, it is still important to understand the nature of the inaccuracies that can arise from GoldSim’s numerical approximations in order to ensure that these do indeed remain small. The primary numerical factors affecting the accuracy of a GoldSim model are as follows:

- **Integrating Differential Equations**: GoldSim solves differential equations by numerically integrating them (via Stocks and Delays). This numerical integration is the largest potential source of numerical inaccuracies in a GoldSim model.

- **Solving Coupled Equations**: In some cases, the system you wish to model may include coupled equations or coupled differential equations. Solutions of these types of equations can be computationally-intensive (e.g., nonlinear coupled differential equations). For some specific types of coupled systems, GoldSim provides fast and accurate solution techniques. In other cases, these equations must be solved approximately using Previous Value elements. The use of Previous Value elements in this case can introduce numerical approximations that are of the same order as those introduced via numerical integration of ordinary differential equations.

  Read more: Using Advanced Algorithms to Solve Coupled Equations (page 1142).

- **Representing Discrete Events**: In many real world systems, discrete events occur which impose discontinuous changes onto the system. Superimposing such discontinuities onto a continuously-varying system discretized in time can introduce inaccuracies. GoldSim provides a powerful timestepping algorithm for accurately representing such systems.

These items are discussed in the topics below.
Note: Round-off error is sometimes noted as an important source of error in simulation models. Although this can indeed be important for some specialized engineering and science simulation tools, given the nature of GoldSim applications, and the fact that GoldSim uses double-precision to carry out its calculations, it is highly unlikely that round-off error could ever have a noticeable impact on a GoldSim model.

Primary Numerical Approximations in GoldSim

Stocks (Integrators, Reservoirs and Pools) represent integrals of the form:

\[ \text{Value} = \text{Initial Value} + \int (\text{Rate of Change}) \, dt \]

The Rate of Change, of course, can be a function of time.

In this case, we are solving the following differential equation:

\[ \frac{d\text{Value}}{dt} = \text{Rate of Change}; \text{Value}_{t=0} = \text{Initial Value} \]

Numerically, GoldSim approximates the integral shown above as a sum:

\[ \text{Value}(t_n) = \text{Initial Value} + \sum_{i=1}^{n} \text{Rate of Change}(t_i - \Delta t_i) \Delta t_i \]

where:

- \( \Delta t_i \) is the timestep length just prior to time \( t_i \) (typically this will be constant in the simulation);
- \( \text{Rate of Change}(t_i - \Delta t_i) \) is the Rate of Change at time \( = t_i - \Delta t_i \), and
- \( \text{Value}(t_i) \) is the value at end of timestep \( i \).

Note that the Value at a given time is a function of the Rate of Change at previous timesteps (but is not a function of the Rate of Change at the current time).

This particular integration method is referred to as Euler integration. It is the simplest and most common method for numerically solving such integrals. The key assumption in the method is that the rate remains constant over a timestep. The validity of this assumption is a function of the length of the timestep and the timescale over which the rate is changing. The assumption is reasonable if the timestep is sufficiently small.

To get a feeling for the errors that can be produced by Euler integration, consider the following very simple integral:

\[ \text{Value} = \text{Initial Value} + \int -k \ast \text{(Value)} \, dt \]

where \( k \) is a constant. This is the equation for simple (exponential) first-order decay, and the analytical solution is:

\[ \text{Value} = \text{Initial Value} \ast e^{-kt} \]

The timescale of the dynamic process can be quantified in terms of a half-life (the time it takes the current value to decay to half of its initial value). The half life is equal to \( 0.693/k \). The table below compares the results of computing the
Value analytically and numerically (by comparing the results at 20 days assuming an initial value of 100 and a half-life of 10 days):

<table>
<thead>
<tr>
<th>Solution</th>
<th>Value at 20 days</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analytical Solution</td>
<td>25.00</td>
<td>-</td>
</tr>
<tr>
<td>Timestep = 5 days</td>
<td>18.24</td>
<td>9.0%</td>
</tr>
<tr>
<td>Timestep = 1 days</td>
<td>23.78</td>
<td>1.6%</td>
</tr>
<tr>
<td>Timestep = 0.5 days</td>
<td>24.40</td>
<td>0.8%</td>
</tr>
<tr>
<td>Timestep = 0.1 days</td>
<td>24.89</td>
<td>0.1%</td>
</tr>
</tbody>
</table>

*Error computed as $\frac{\text{Simulated - Analytical}}{\text{Analytical - Initial Value}}$.

A plot of these results is shown below:

As can be seen, with the exception of the 5 day timestep, the Euler integration method is relatively accurate. In fact, for most systems that you will be simulating using GoldSim, a numerical error of several percent is likely to be acceptable (and much smaller than the error caused by the uncertainty in the initial conditions, the parameters, and the conceptual model). As a general rule, your timestep should be $1/3$ to $1/10$ of the timescale of the fastest process being simulated in your model.

**Read more:** [Selecting the Proper Timestep](page 1140).

**Warning:** The magnitude of the integration error depends on the nature of the model. In stable models that are dominated by negative feedback loops (and hence tend toward equilibrium), the errors tend to diminish with time. Systems that are unstable and grow exponentially or oscillate with no damping tend to accumulate errors over time. For these types of systems (e.g., a swinging pendulum), a very small timestep may be required to accurately simulate the system using Euler integration.
**Note:** In cases where a small timestep is required to maintain accuracy, this can be done in a very computational efficient way by using Containers with Internal Clocks, which allow you to locally use a much smaller timestep.

**Other Integration Methods**

Although the Euler method is simple and commonly used, other integration methods exist which can achieve higher accuracy with a larger timestep (e.g., Runge-Kutta, variable timestep methods). Because these methods can use a much larger timestep without losing accuracy, they are more computationally efficient.

These methods, while being important for some types of systems (e.g., sustained oscillators like a pendulum), are not incorporated into GoldSim for the following reasons:

- Higher order methods are most useful when simulating physical systems in which the mathematical model, initial conditions and input parameters are known very precisely, and small integration errors can be significant. For the most part, the kinds of systems that you will be simulating using GoldSim can be handled effectively using Euler integration.

- Higher-order methods work best when simulating continuously-varying systems. They do not deal well with systems that behave discontinuously and/or are impacted by discrete changes. Most real world systems do not vary continuously, and GoldSim therefore provides powerful algorithms for accurately superimposing discrete changes (discontinuities) onto a continuously-varying system. These algorithms are incompatible with higher-order integration methods.

  **Read more:** [Accurately Simulating Discrete Events that Occur Between Timesteps](#) (page 1141).

- For some kinds of systems (e.g., mass or heat transport using the Contaminant Transport Module), GoldSim uses powerful algorithms to accurately solve nonlinear coupled differential equations. These algorithms are incompatible with higher-order integration methods.

  **Read more:** [Using Advanced Algorithms to Solve Coupled Equations](#) (page 1142).

As mentioned above, in cases where a small timestep is required to maintain accuracy using Euler integration, this can be done in a very computational efficient way by using Containers with Internal Clocks, which allow you to locally use a much smaller timestep.

**Approximate Solutions to Coupled Equations**

In some situations, you may wish to simulate a static or dynamic processes in which the variables in the system are coupled such that they respond instantaneously to each other, with no time lags. In GoldSim, these are referred to as *recursive loops*, and they are treated differently from feedback loops.

If you encounter a system such as this in one of your models, you can handle it in one of two ways. First, you could solve the system of equations directly either prior to running the model (e.g., using substitution), or dynamically while running the model (e.g., using an External element or GoldSim’s matrix functions to solve the appropriate equations). Note, however, that for many
Summary of GoldSim’s Dynamic Timestepping Algorithm

complex models (e.g., non-linear coupled equations), the solution could be very computationally intensive.

GoldSim offers an alternative: solve the equations approximately and iteratively using Previous Value elements. A Previous Value element allows you to reference the previous value (i.e., the previous timestep) of an output.

Read more: Creating Recursive Loops Using Previous Value Elements (page 960).

As a general rule, your timestep should be 3 to 10 times shorter than the timescale of the fastest process being simulated in your model. In simple systems, you should be able to determine the timescales of the processes involved. In more complex models, however, it may be difficult to determine the timescales of all the processes involved.

In addition, for some kinds of models (e.g., some oscillating systems), this general rule may not be sufficient and you may need a smaller timestep).

Therefore, after building your model, you should carry out the following experiment:

1. Carry out an expected value or median value simulation.
2. Reduce the timestep length by half (increase the number of timesteps by a factor of 2), rerun the model, and compare results.
3. Continue to half the timestep length until the differences between successive simulations are acceptable.

Summary of GoldSim’s Dynamic Timestepping Algorithm

GoldSim provides a powerful timestepping algorithm that can dynamically adjust to more accurately represent discrete events, respond to rapidly changing variables in your model, represent specified SubSystems in your model using different timesteps, and accurately represent some special kinds of systems (e.g., mass and heat transport within the Contaminant Transport Module).

These features are discussed in detail elsewhere in this document. To complement the rest of the information provided in this appendix, however, these features are summarized here.

GoldSim allows you to increase or decrease the timestep length according to a specified schedule during a simulation (e.g., start with a small timestep, and then telescope out to a larger timestep). This can be useful, for example, if you know that early in a simulation, parameters are changing rapidly, and hence you need a smaller timestep.

You do this by defining Periods Steps, which have different durations and timestep lengths. An example of pre-specified time periods is shown below:

In this example, the model uses a 0.2 day timestep for the first 10 days, a 1 day timestep between 10 days and 20 days, and a 5 day timestep between 20 days and 50 days (Although not indicated here, it then reverts to the default timestep of 10 days for the remainder of the simulation).
Read more: Adding Shorter Timesteps Over Defined Periods (page 467).

Although defining shorter timesteps over defined periods can be very useful, you must fully specify them prior to running the simulation. That is, you must know how you would like to alter your timestep prior to running the model. In some cases, however, it may not be possible to do this. That is, in complex systems (particularly ones with uncertain parameters), variables may change at different rates in different realizations, in ways that you cannot predict prior to running the model.

To better simulate these kinds of systems, GoldSim provides an advanced feature that allows you to dynamically adjust the timestep during a simulation (i.e., insert “internal” timesteps) based on the values of specified parameters in your model. For example, you could instruct GoldSim to use a timestep of 1 day if \( X \) was greater than \( Y \), and 10 days if \( X \) was less than or equal to \( Y \). Similarly, you could instruct GoldSim to use a short timestep for a period of 10 days after a particular event occurs, and then return to the default timestep.

Read more: Dynamically Controlling the Timestep (page 471).

In addition to providing a dynamic timestepping algorithm on a global scale (i.e., for the entire model), GoldSim also enables you to apply dynamic timestepping to specific Containers. This allows you to specify different timesteps for different parts (i.e., Containers) in your model. For example, if one part of your model represented dynamics that changed very rapidly (requiring a 1 day timestep), while the rest of the model represented dynamics that changed much more slowly (requiring a 10 day timestep), you could assign a 10 day timestep to the model globally, and a 1 day timestep to the container representing the SubSystem that changed rapidly.

Read more: Specifying Containers with Internal Clocks (page 474).

In some cases, events or other changes in the model may not fall exactly on a scheduled update. That is, some events or changes may actually occur between scheduled updates of the model. These trigger an “unscheduled update” of the model. Unscheduled updates are timesteps that are dynamically inserted by GoldSim during the simulation in order to more accurately simulate the system. That is, they are not specified directly prior to running the model. GoldSim inserts them automatically (and, generally, without you needing to be aware of it).

“Unscheduled updates” can be generated in the following ways:

- When events are output by a Timed Event, Event Delay, Discrete Change Delay or Time Series element;
- By manually specifying a dynamic timestep (i.e., dynamically controlling the time between updates);
- When a Reservoir element reaches an upper or lower bound;
- When a Resource becomes exhausted;
- When any element is triggered by an At Stock Test, At Date or At Etime triggering event; and
When any of these events occur, GoldSim automatically inserts an unscheduled update at the exact time that the event or change occurs. For example, if you had specified a one day timestep, and a Timed Event occurs at 33.65 days (i.e., between the scheduled one-day updates), GoldSim would insert an unscheduled update at 33.65 days.

Read more: Understanding Timestepping in GoldSim (page 455).

By default, scheduled updates are always dynamically inserted by GoldSim. However, in some (rare) cases, you may want to prevent unscheduled updates from being inserted. For example, if your model included a specialized algorithm that was designed based on the assumption that the timestep was constant, inserting unscheduled updates could invalidate the algorithm. To support such situations, GoldSim allows you to disable unscheduled updates.

Warning: Because unscheduled updates are intended to more accurately represent a complex dynamic system, disabling this feature should be done with caution, and is generally not recommended.

Read more: Controlling Unscheduled Updates (page 470).

For some special types of systems, GoldSim provides additional dynamic timestepping algorithms (different from the timestep algorithms described above) to more accurately solve these equations. For example, the Contaminant Transport Module utilizes dynamic timestep adjustment to solve the coupled differential equations associated with mass and heat transport.

This algorithm allows GoldSim to handle “stiff” systems (systems with widely varying time constants) as well as nonlinear aspects of the system in a very accurate and computationally efficient manner. This algorithm is discussed in the GoldSim Contaminant Transport Module User’s Guide.
Glossary of Terms

Activation ID
A 32-digit code (8 groups of 4 digits each, separated by hyphens) that is used to activate your license.

Active Scenario
When scenarios have been defined, the scenario that is being viewed when you are browsing a model.

Affects View
A special browser view in GoldSim that allows you to see all the elements the selected element affects.

Alias
The name by which an exposed output of a localized Container is referenced.

Allocator
An element that allocates an incoming signal to a number of outputs according to a specified set of demands and priorities. Typically, the signal will be a flow of material (e.g., water), but it could also be a resource, or a discrete transaction.

Array
A collection of variables that share common output attributes and can be manipulated in GoldSim elements or input expressions.

Array Labels
A collection of labels identifying the items of an array.

Autocorrelate
To correlate a Stochastic element to a previous value of itself.

Browser
An alternative view of a GoldSim model, in which elements are displayed in a tree, and organized either hierarchically, or by type.

Built-in Constants
Constants (such as pi) that are built-in to GoldSim and can be used when creating expressions in input fields.

Built-in Functions
Mathematical functions (such as sine, maximum, round) that are built-in to GoldSim and can be used when creating expressions in input fields.

Causality Sequence
The specific order in which GoldSim updates (computes) elements every timestep.
Change Note
A note added to an element when using versioning that is subsequently displayed when showing changes.

Chart Style
A collection of settings for a particular type of result display chart.

Clones
Sets of elements whose properties change simultaneously when any one member of the set is edited.

Complementary Cumulative Distribution Function
The complement of the cumulative distribution function.

Conceptual Model
A representation of the significant features, events and processes controlling the behavior of a system.

Condition
An output Type. A Condition is a Boolean switch (e.g., Yes/No, True/False, On/Off).

Conditional Expression
An expression which evaluates to (produces) a Condition (rather than a Value).

Conditional Tail Expectation
The expected value of the output given that it lies above a specified quantile. That is, it represents the mean of the worst $100(1 - \alpha)\%$ of outcomes, where $\alpha$ is the specified quantile.

Container
An element that acts like a "box" or a "folder" into which other elements can be placed. It can be used to create hierarchical models.

Convolution Element
An element that solves a convolution integral.

Coupled Link
A link between two elements which causes the elements to be solved in a coupled (rather than a sequential) manner. Coupled links can only be created when using an extension module.

Cumulative Distribution Function
The integral of a probability density function.

Data Source
A source of data external to your GoldSim model that can be automatically imported into GoldSim elements. External data sources are either spreadsheets, text files, databases or DLLs.

Date-time Simulation
A simulation that tracks time using the simulated Date/time.
Delay Elements
A class of elements that simulate processes that delay continuous or discrete signals and flows. The output of a delay element lags its inputs.

Deterministic Simulation
A simulation in which the input parameters are represented using single values (i.e., they are "determined" or assumed to be known with certainty).

Dimensions
An output attribute for an element that defines the dimensionality (in terms of Length, Time and other fundamental dimensions) of the output.

Discrete Change
An element that generates discrete change signals that can subsequently modify stock elements.

Discrete Change Signal
A discrete signal that contains information regarding the response to an event.

Discrete Event Signal
A discrete signal indicating that something (e.g., an accident, an earthquake, a bank deposit) has occurred.

Discrete Signal
A special category of output that emits information discretely, rather than continuously.

Display Units
The units (e.g., m, g, $/day) in which an output is displayed within GoldSim.

Drawing Tools Toolbar
A toolbar at the top of the GoldSim interface that provides buttons for adding graphical components to your model.

Edit Mode
The state of a model when it is being edited and does not contain simulation results.

Elapsed Time Simulation
A simulation that tracks time using the elapsed time.

Euler Integration
A simple and commonly used numerical integration method.

Exposed
Description of an output within a localized Container that can be referenced outside of the Container.

Expression Element
A function element that produces a single output by calculating user-specified mathematical expressions.
External Functions
User-defined modules that can be linked to GoldSim at runtime as Dynamic Link Libraries (DLLs).

Feedback Loop
A looping system in which the variables in the loop represent a closed chain of cause and effect. Note that the terms “feedback” and “cause and effect” intentionally imply that the relationship between the variables is dynamic and the system changes over time (although systems with feedback loops can also reach a dynamic equilibrium). Feedback loops contain at least one state variable.

Function Elements
Elements that instantaneously compute outputs based on one or more defined inputs.

Function Of View
A special browser view in GoldSim that allows you to see all the elements that affect the selected element.

Graphical Objects
Objects in GoldSim that are used to embellish or document the model.

Graphics Pane
The primary portion of the GoldSim interface, where the graphical depiction of the model is shown.

History Generator
An element that generates stochastic time histories of variables. A stochastic time history is a random time history that is generated according to a specified set of statistics.

Importance Sampling
An algorithm that biases sampling of probability distributions in order to better resolve the tails of the distributions.

Information Delay
A delay element that delays information signals, and does not enforce conservation of the signal.

Input Elements
Elements that are used to define basic inputs for a model.

Interactive Result
A result display that is shown in a modal window (i.e., windows that always retain the focus). Interactive results can be converted to modeless Result elements.

Keywords
Text delimited by the % symbol (e.g., %x_unit%) that can be used when creating chart styles to insert context sensitive text (e.g., for axis labels).
Kurtosis
A measure of how "fat" a distribution is relative to a normal distribution with the same standard deviation. A normal distribution has a kurtosis of 0. The kurtosis is a function of the fourth moment of the distribution.

Latin Hypercube Sampling
A stratified sampling method that has the effect of better ensuring that the space of the parameter is uniformly spanned.

Link Cursor
A special cursor for creating links invoked by double-clicking on an input or output object in a browser.

Live Model
When using GoldSim’s scenario features, a “scratch” model, or a temporary placeholder model where you can experiment before saving something as a scenario.

Localize
An action that you can apply to a Container that creates a separate scope for the elements in that Container.

Lookup Table Element
A function element that allows you to create a 1, 2, or 3-dimensional lookup table (response surface).

Material
Tangible things (water, dirt, cash, widgets) that are tracked in a simulation.

Material Delay
A delay element that delays flows of materials (e.g., masses, volumes, items).

Mathematical Model
A set of input assumptions, equations and algorithms describing the behavior of a system.

Matrix
A two-dimensional array.

Mean
The expected value (average) of a distribution. It is the first moment of the distribution.

Median
The 50th percentile of a distribution.

Menu Bar
A bar at the top of the GoldSim interface that provides access to menus from which nearly any GoldSim operation can be carried out.

Model Objects
Objects in GoldSim that are used to quantitatively represent the variables and relationships in a model. The primary model object is the element.
Model Root
The top-level Container in a GoldSim model.

Modes
The states that a GoldSim model can be in at a given time. There are three modes: Edit Mode, Run Mode, and Result Mode.

Monte Carlo Simulation
A method for propagating (translating) uncertainties in model inputs into uncertainties in model results.

Network License
A GoldSim license that resides on a License Server and is intended to be shared between multiple users. In order to use the license, your computer must be persistently connected to the License Server (although Network licenses can be “borrowed” from the License Server so that they temporarily reside on your computer and a connection to the License Server is no longer necessary).

Normal Link
A standard link between two elements.

Note Pane
A dockable window in GoldSim that displays a Note associated with the selected element.

Numerical Integration
An approximate solution to an integral equation, carried out by discretizing a variable (e.g., time) into discrete intervals.

Output Attributes
Three properties of an element’s output that determine the kinds of inputs to which it can be linked: type, order and dimensions.

Performance Measure
A specific model output by which you judge the performance of a system.

Plot Points
Points in time at which the value of outputs are saved for plotting time history data.

Pool
A stock element that integrates and conserves flows of materials. A Pool is a more powerful version of a Reservoir (it has additional features to more easily accommodate multiple inflows and outflows).

Ports
Small boxes on the side of the element in the graphics pane. You can left-click on a port to access the element’s inputs and outputs.

Precedence
The order in which mathematical operators are evaluated in an expression.
**Previous Value Element**
An element that outputs the value of its input from the previous model update.

**Primary Output**
For an element with multiple outputs, the output that has the same name as the element.

**Probabilistic Simulation**
A simulation in which the uncertainty in input parameters is explicitly represented by defining them as probability distributions.

**Probability Density Function**
A plot of the relative likelihood of the values of an uncertain variable.

**Probability Distribution**
A mathematical representation of the relative likelihood of a variable having certain specific values.

**Probability Histories**
A probabilistic representation of the time history of an output in which the percentiles for multiple realizations are plotted.

**Probability Mass Function**
A plot of the relative likelihood of the values of a discrete uncertain variable.

**Realization**
A single model run within a Monte Carlo simulation. It represents one possible path the system could follow through time.

**Recursive Loop**
A system with circular logic that does not contain any state variables.

**Reference Version**
The version to which your current model is compared when tracking changes.

**Reporting Periods**
Regular time points during a simulation (e.g., every month, every year) at which you can compute and view results associated with that period (e.g., monthly averages, annual cumulative values).

**Reservoir**
A stock element that integrates and conserves flows of materials.

**Resource**
Something that has a limited supply (e.g., spare parts, fuel, skilled personnel, money) and is required in order for elements of the modeled system to carry out certain actions.

**Result Element**
An element that can be used to organize, analyze and display results.
**Result Mode**
The state of a model when it has been run and contains simulation results for a single set of input parameters.

**Run Control Toolbar**
A toolbar at the top of the GoldSim interface that provides buttons for running a model.

**Run Log**
Text that is stored with a GoldSim model once it has been run. It contains basic information regarding the simulation, and any warning or error messages that were generated.

**Run Mode**
The state of a model when it is running.

**Run Properties**
A set of fundamental properties that track the progress of the simulation (e.g., Time, Realization) and can be referenced like outputs in expressions.

**Scalar**
An output consisting of a single value or condition.

**Scenario**
A specific set of input data (and corresponding outputs) for a model. Multiple scenarios can be defined for a model. Different scenarios within a model are specifically differentiated by having different values for one or more Data elements.

**Scenario Data**
Data elements that differentiate the various scenarios in a model.

**Scenario Manager**
A dialog that allows you to create, define and run scenarios.

**Scenario Mode**
The state of a model when it contains scenario results, allowing multiple scenarios to be compared.

**Scheduled Timesteps**
Timesteps that are directly specified by the user prior to running the model.

**Scope**
The portion of a model from which an element’s output can be referenced. You cannot reference an element in a different scope unless that output is specifically exposed.

**Script Element**
An element that can be used to create a list of executable statements (e.g., variable definitions, variable assignments and statements controlling the sequence of execution such as loops and if statements) in order to implement complex logic and operations.
Secondary Output
For an element with multiple outputs, an output that has a different name than the element.

Simulation Model
The implementation of a mathematical model of a system within a specific computational tool (or set of tools).

Skewness
A measure of the symmetry of a distribution. A symmetric distribution has a skewness of 0. The skewness is a function of the third moment of the distribution.

Splitter
An element that splits an incoming signal between a number of outputs based on specified fractions or amounts. Typically, the signal will be a flow of material (e.g., water), but it could also be a resource, or a discrete transaction.

Spreadsheet Element
An element that can dynamically link to an Excel spreadsheet.

Standard Deviation
The square root of the variance of a distribution. The variance is the second moment of the distribution and reflects the amount of spread or dispersion in the distribution.

Standalone License
A GoldSim license that resides on an individual computer and is licensed for use just on that computer. As such, it can be used by only one person at a time. The license can be transferred between computers (quickly and easily). There are two types of Standalone licenses: Desktop Standalone licenses, and Enterprise Standalone licenses. The licenses are identical in all respects with one exception: Desktop Standalone licenses limit the number of license transfers (to 6 per year). Enterprise Standalone licenses allow an unlimited number of transfers.

Standard Toolbar
A toolbar at the top of the GoldSim interface that provides buttons for common GoldSim actions.

State Variable
The output of an element in GoldSim whose value is computed based on the historical value of the element’s inputs (as opposed to only being a function of the current value of the element’s inputs). State variables have well-defined initial conditions. Feedback loops can only be created if they contain at least one state variable.

Status Bar
A bar at the bottom of the GoldSim interface that provides information regarding the status of the model.

Stochastic
An element that can be used to quantitatively represent the uncertainty in a model input.
Stock Elements
A class of elements that numerically integrate inputs, and hence are responsible for internally generating the dynamic behavior of many systems.

Store
Stockpiles or places where a Resource (e.g., parts, personnel) is stored or located when not being used. Resource Stores can be thought of as having physical locations in the system you are modeling. The can be global or local (associated with a Container).

Style File
An external file which stores chart style that you can import.

SubModel
A specialized element that allows you embed one complete GoldSim model within another GoldSim model. This facilitates, among other things, probabilistic optimization, explicit separation of uncertainty from variability, and manipulation of Monte Carlo statistics.

SubSystem
A specialized Container that is completely “self-contained”. SubSystems can take on some useful features and properties (e.g., conditionality, having an internal clock, and being able to loop), but also have some limitations (with regard to how they can be incorporated into feedback loops).

System Units File
A file on your hard drive (named units.dat) that stores all of your unit settings.

Table Function
A special function for referencing user-defined lookup tables that can be referenced in input fields. It is automatically created whenever you create a Lookup Table element.

Timed Event
An element that generates discrete event signals based on a specified rate of occurrence.

Timestep
A discrete interval of time used in dynamic simulations.

Total System Model
A simulation model that focuses on creating a consistent framework in which all aspects of the system, as well as the complex interactions and interdependencies between subsystems, can be represented.

Unit Categories
A category of units defined by a name (e.g., Area) and a specific set of dimensions (Length^2).

Unit Strings
Strings containing only unit abbreviations used to define compound units (e.g., m/sec, lbf/m2).
**Unscheduled Updates**
Timesteps that are inserted automatically by GoldSim during a simulation and are not directly specified by the user prior to running the model.

**Vector**
A one-dimensional array.

**Version**
A "snapshot" of your model at a particular point in time.

**Versioning**
The process of tracking changes that you make to your model file.
Index

.GSM extension 62

2
2D scatter plots 677

3
3D scatter plots
described 679

A
Aborting a run 503
Absolute value function 125
Accuracy of simulation models 1136
Activating conditional Containers 904
extension modules 18
Standalone license 8
Activating GoldSim 7
Activation event
output of conditional Container 904
Active Scenario 509
Activity status
output of conditional Container 903
Adding
graphic objects to graphics pane 743
hyperlinks in graphics pane 760
hyperlinks in Notes 759
hyperlinks in text boxes 753
images to graphics pane 753
multiple outputs to a distribution display 654
multiple outputs to time history display 588
new elements 73
nodes to influences 427
notes to elements 758
outputs to a multi-variate display 672
Result elements 571
shapes to graphics pane 743
text boxes to graphics pane 751
text to an influence 428
text to graphics pane 747
Affects View 112
SubModels 1009
Aging chains
overview 810
Aliases for exposed outputs 949
Aligning objects 767
Alignment of timestep 459
Allocator elements 284
Analysis description 485
And elements 291
Annuity, computing 128
Appearance
of charts 702
of elements 434
of graphic objects 746
of hyperlinks 763
of influences 426
of text boxes 752
of text objects 748
Appendices, list of 6
Arcosine function 124
Archiving files
with results 497
Arcsine function 124
Arctangent function 124
Array labels
creating 784
deleting 789
understanding 783
Array results 689
controlling chart style 700
defined 562
displaying conditions 697
screening results 699
viewing a matrix chart 694
viewing a vector chart 692
viewing multiple realizations 698
viewing properties 690
viewing tables 697
Arrays
copying between models 807
creating with constructor functions 795
creating with Data elements 790
creating with Stochastic elements 795
defining and assigning in scripts 870
displaying time histories 592
functions that operate on 801
introduction 41
manipulating in expressions 799
manipulating with elements 806
plotting time histories of multiple realizations 606
referencing an item 793
using 782
using as lookup tables 798
viewing in browsers 793
Arrows between elements (influences) 98
Auto triggers in Conditional Containers 907
Autocorrelate 179
Autocorrelating Stochastics 179
Automatic triggering 358
Auto-save 63
Auto-suggesting input links 83
Axes in charts 705

Background
changing for element 437
in graphics pane 423
Basic step length 459
Batch runs 549
Bayesian updating 1016
Bessel function 126
Beta distribution 159, 160, 1056
Beta function 126
BetaPERT distribution 161
Binomial distribution 163, 1057
Boolean distribution 163, 1057
Bounds
for Pools 258
for Reservoirs 240
Braces, to identify units 91
Brackets, to reference array items 793
Browser
activating 105, 755
context menu for 109
deactivating 105, 755
described 59
display tool-tips 61
docking 105, 755
hiding 105, 755
moving location 105, 755
Search field 112
synchronizing with graphics pane 108
types of views 107
using 105
viewing arrays 793
Browser
Previous and Home buttons 86
Browser button 105, 755
Built-in constants 129
Built-in functions 124
financial 128
math 125
special 126
trigonometry 124
Buttons
in toolbars 59

C
Calculation logic 340
Calendar Time simulation 454
Capture points 469
Capture points for timesteps viewing results at 569
Casting units 93
Categories of elements 146
Causality sequence 343, 968
ambiguous 349
invalid 349
modifying 971
viewing 968
CCDF 1046
CDF 1045
computing 1084
Ceiling function 125
Change control 1023
Change Note 1029
Changed function 126
Changing
element appearance 434
element background and outline 437
element labels 436
element symbols 434
graphic object appearance 746
image appearance in graphics pane 754
influence appearance 426
size of graphics pane 425
text appearance in graphics pane 748
text box appearance in graphics pane 752
Chart styles
applying 712
array results 700
axes 705
defining defaults 716
distribution results 668
editing 702
General tab 703
grid 709
header and footer 704
importing and exporting 715
introduction 581

1156 • Index
Index

- keywords 717
- legend 709
- managing using style manager 715
- multi-variate results 688
- saving 712
- style manager 714
- time history results 635
- Z-axis 708
- Charts 564
  - context-sensitive menu 566
  - copying 734
  - distributions 647
  - editing appearance 702
  - exporting 734
  - styles 581
  - time histories 584
  - tool-tips 567
  - zooming within 567
- Circular systems
  - feedback loops 346
  - recursive loops 960
- Class View 107
- Classifying realizations 577
- Clones
  - Containers 955
  - copying 954
  - copying Containers with clones 956
  - creating 953
  - freeing 955
  - moving 954
  - understanding 952
- Closed curves, adding to graphics pane 745
- Closing files 62
- Coefficient of determination computing 1090
- Coefficient of variation of a distribution 1047
- col
  - use in array constructors 795
- Command line, running model from 549
- Complementary cumulative distribution function 1046
- Completion event
  - output of conditional Container 904
- Completion status
  - output of conditional Container 903
  - output of Milestone 403
- Conceptual model 25
- Cond. Tail Expectation
  - for distribution results 644
  - for Stochastic elements 157
  - function 182
- Conditional Containers
  - activating 904
  - activation event 904
  - activity status output 903
  - auto triggers 907
  - behavior of elements within 899
  - browser view 909
  - completion event 904
  - completion status output 903
  - creating 135, 901
  - deactivating 905
  - duration output 903
  - Number of Activations output 903
  - outputs 903
  - specifying resources 907
  - termination event 904
  - using 899
- Conditional Tail Expectation computing 1088
- Conditional Tail Expectation for distribution results 644
  - for Stochastic elements 157
  - function 182
- Conditions
  - defining outputs as 87
  - displaying in array charts 697
  - displaying in distribution charts 650
  - displaying in time history charts 585
  - expressions 130
- Confidence bounds on distributions 645, 648, 1086
- Confidence bounds on the mean 1086
- Constants 129
- Constructor functions for arrays 795
- Containers
  - appearance of contents 139
  - cloning 955
  - conditional 135, 899
  - copying 102
  - copying clones 956
  - features 133
  - global or local 134
  - influences between 99
  - internal clocks 136
  - localized 944
  - locking 144
  - looping 137
  - moving elements between 102
  - navigating within 96
overview 95
properties dialog 132
protecting 139
protecting contents from editing 144
providing Resources 138
sealing 143
SubSystems 135
summary information 140
using 132
using to disable Time History
Result elements 141
using to save results 141
viewing in browser 107
Containment View 107
Contaminant Transport Module 53
Context-sensitive Help 21
Context-sensitive menu
displaying 61
in browser 109
in charts 566
in input fields 85
using to insert elements 73
Conventions
to describe key combinations 7
to describe mouse actions 7
Conveyor-belt
treating Discrete Change Delay as 394
treating Event Delay as 381
Convolution elements 817
defining as arrays 821
defining inputs 818
examples 821
theory 817
Copula algorithms for correlation 1070
Copying
charts and tables 734
Containers 102
data into a 1-D Table 297
data into a 2-D Table 300
data into a 3-D Table 304
data into a Time Series 197
data into an array 793
elements 101
elements between files 103
graphic objects 770
copying table results 567
Correlating elements
algorithm 1070
Correlating Stochastics 178
Correlation
for History Generator elements 837
for Stochastic vectors 185
Correlation algorithms 1070
Correlation coefficients
computing 1090
Correlation matrix
viewing 683
Cosine function 124
Cotangent function 124
Creating
array labels 784
arrays using constructor functions 795
arrays using Data elements 790
arrays using Stochastic elements 795
clones 953
conditional Containers 135, 901
Extended GoldSim database 1128
Generic database 1122
hyperlinks in graphics pane 760
hyperlinks in Notes 759
hyperlinks in text boxes 753
links 80, 95
model versions 1024
notes 758
Result elements 571
Simple GoldSim database 1122
units 441
units for items (e.g., widgets) 444
Cumulative distribution 164, 1058
Cumulative distribution function 1045
computing 1084
Current service time for an Event Delay 379
Curves, adding to graphics pane 745
Custom statistics for time histories 603
Customizing
the graphics pane 423
timesteps 466
D
Dashboard Authoring Tools 54
Dashboards
default 778
Data display
multi-variate results 684
Data elements 151
using to create arrays 790
Data source
linking to Lookup Tables 305
Data styles 710
Data tips in charts 567
Database links 1031
adding data sources 1033
  downloading definitions 1034
  downloading from Extended GoldSim 1038
  downloading from Generic 1035
downloading from Simple 1037
globally downloading 1040
removing 1041
to Lookup Tables 311
types 1032
validating 1035

Dates
  Date and Datetime units 445
  referencing in expressions 131
  using in Spreadsheet elements 930
  using in Time Series 197
  DateTime 489
  Date-time simulation 454

Deactivating
  conditional Containers 905
  extension modules 18
  link cursor 95
  Standalone license 10

Deadband
  representing using a Status element 400

Decision elements 370
Default Dashboard 778
Delay elements 72, 149, 320
Deleting
  elements 101
  graphic objects 744
  influences 87
  links 87
  notes 758, 759
  text 748
text boxes 751

Derivative of Lookup Table 316
Description field for elements 79
Desktop Standalone license 8

Deterministic simulations
  compared to probabilistic 1051
  defining values for Stochastics 180
  running 482

Dimensions 89
Disabling a Result element 635
Discrete Change Delay elements 390
  conveyer-belt approach 394
  no dispersion 393
  Number in queue 396
  referencing the discrete change Value 395
  simulating queues 396

simulating queues using
  Resources 396
  specifying capacities 396
  specifying Resources 396
time-variable delay times 394
  with dispersion 393
Discrete Change elements 383
Discrete change signals
  defined 353
  generating 281, 284, 383, 388, 390, 415
  routing using Splitter 397

Discrete changes
  generating with Time Series elements 415
Discrete Changes
to a Pool 388
to a Reservoir 386
to an Integrator 385
Discrete distribution 165, 650, 1059
Discrete event signals
  defined 353
  generating 364
Discrete events 351
Discrete outputs 89
Discrete signals 89, 353
Disruption
  Discrete Change Delays 393
  Event Delays 379
  Information Delays 324
  Material Delays 332
  Display units 89

Displaying
  CDFs 1084
  confidence bounds on distributions 1086
  confidence bounds on the mean 1086
  notes 758
  PDFs 1085
toolbars 422
tool-tips 61, 109

Displaying results
  arrays 689
distributions 639
  multiple distributions in one chart 654
  multiple time histories in one chart 588
  overview 556
  single realization distribution displays 658
time histories 581
while model is running 574
Distributed Processing Module 55
Distribution results 639
classifying results 658
controlling chart style 668
defined 559
displaying conditions 650
displaying discrete results 650
properties 640
result array 653, 1084
screening results 658
viewing a distribution summary 643
viewing charts 647
viewing Distribution outputs 663
viewing multiple outputs 654
viewing scenarios 665
viewing single realizations 658
viewing tables 651

Distributions
beta 159, 160
Beta 1056
BetaPERT 161
binomial 162, 1057
Boolean 163, 1057
correlated 1048
cumulative 164, 1058
discrete 165, 1059
editing 155
exponential 166, 1060
externally-defined 167
extreme probability 168, 1060
extreme value 168, 1061
gamma 169
Gamma 1061
log-cumulative 1058
log-normal 170, 1062
log-triangular 1067
log-uniform 1068
mathematics of 1056
moments of 1046
negative binomial 1063
negative binomial 170
normal 171, 1063
Pareto 172, 1064
Pearson Type III 172, 1064
Poisson 173, 1065
Sampled result 1065
sampled results 174
specifying 1047
Student’s t 174, 1065
triangular 175, 1066
truncated 1069
understanding 1044
uniform 176, 1068
Weibull 1069
Weilbull 176

Divide by zero in input fields 83
DLLs
calling details 1107
calling from GoldSim 931
defining Lookup Tables using
312
details 1096
elements 1102
running in a separate process 937
Docking toolbars and menu bars 423
Documenting models
described 737
overview 49
Drawing Tools toolbar 742
Duration
output of conditional Container 903
Dynamic simulation 24, 34
Dynamically adjusted timesteps 471

E
EDay 491
Edit Mode 498
Editing
array labels 784
chart appearance 702
element appearance 434
element properties 77
graphic objects 746
Hyperlinks 761
influence appearance 426
notes 759
text boxes in graphics pane 752
text in graphics pane 748
EHour 491
Elapsed time
referencing 101, 486
run properties 490
Elapsed Time simulation 454
Element toolbar 73
Elements
Allocator 284
And 291
categories 30, 146
changing appearance of 434
changing background 437
changing labels 436
changing outline 437
cloning 952
Containers 95
Convolution 817
copying 101
copying between files 103
creating 67, 73
Data 151
Decision 370
defined 29
Delay 72, 149, 320
deleting 101
dependencies 112
description 79
Discrete Change 383
Discrete Change Delay 390
ing editing properties 61, 77
Event 71, 148
Event Delay 377
Expression 275
External 931
Extrema 276
File 941
finding 112
Function 69, 147, 274
History Generator 826
ID 78
images for 434
in conditional Containers 899
Information Delay 321
Input 68
Inputs 146
inputs and outputs 74
inserting new 73
Integrator 228
Interrupt 409
linking to databases 1031
links between containers 99
Logical 290
Lookup Table 294
Material Delay 329
Milestone 403
moving between containers 102
naming conventions 78
Not 293
Or 292
overview of 29
overview of categories 146
Pool 252
Previous Value 957
Random Choice 373
Reservoir 235
resetting images for multiple
  elements 435
resetting multiple 435
Result 72, 149, 570
Script 860
selecting with mouse 60
Selector 278
Splitter 281
Spreadsheet 910
Status 398
Stochastic 153
Stock 69, 147, 228
Sum 289
symbols for 434
Time Series 186
Timed Event 365
tool-tips for 109
Triggered Event 368
types of 30, 68, 146
using to manipulate arrays 806
Ellipses, adding to graphics pane 745
Email
  sending a model via 66
  support 21
Embedding models within models 973
emf, creating in GoldSim 435
EMinute 491
EMonth 491
Empirical expressions 93
Enabling/disabling result elements 141
Enhanced metafile, creating in
  GoldSim 435
Enterprise Standalone license 8
equality and inequality
  precision 82
EQuarter 491
Error function 126
ESecond 491
ETime 490
Euler integration 35, 230, 1137
Event Delay elements 377
couveyor-belt approach 381
Current service time 379
Mean time 379
no dispersion 379
Number in queue 381
simulating queues 381
simulating queues using
  Resources 382
specifying capacities 381
specifying Resources 382
time-variable delay times 381
  with dispersion 379
Events
  automatic triggering 358
  basic concepts 352
calculation sequence 418
causality sequence 418
determining if they occur 417
Elements for simulating 71, 148
  propagating between elements 353
  queuing 381
Index

responding to 383
simulating 351
triggering 355
Example models 5
Exponential distribution 166, 1060
Exponential function 125
Exponential smoothing 328
Exporting
  chart styles 715
  charts 734
  for statistical post-processing 733
  graphics pane 772
  multi-variate results 733
  results 719
  results to spreadsheets using
    Result elements 720
  results to spreadsheets using
    Spreadsheet elements 732
  results to text files using Result elements 727
  XML model inventory 772
Exposing outputs on a Container 946
Expression elements 275
Expressions
  conditional 130
  empirical 93
  entering and editing 81
  mathematical operator precedence 81
  referencing dates 131
  referencing time 101, 486
  relational operator precedence 130
Extended GoldSim database
  creating 1128
  downloading from 1038
Extension modules
  activating and deactivating 18
  overview 52
External elements 931
  controlling when DLL is called 940
  defining lookup table outputs 937
  details 1096
  inputs and outputs 932
  locking onto a file 936
  reading and creating time series 939
  running DLLs in a separate process 937
  saving outputs 941
  steps to create 932
External functions
  calling details 1107
  calling sequence 1105
  examples 1102
  implementing 1096
Externally-Defined distribution 167
Extrema elements 276
Extreme probability distribution 1060
Extreme Probability distribution 168
Extreme value distribution 1061
Extreme Value distribution 168
EYear 491

F
Feedback loops
  finding 348
  how evaluated 346
  involving SubSystems 348
  types 347
File elements
  described 941
  locking onto a file 943
  to access a network file 943
  to support distributed processing 942
Files
  archiving 62
  archiving with results 497
  auto-save 63
  closing 62
  example models 5
  extension GSM 62
  opening 62
  Player 777
  protecting 65
  recovering 63
  saving 62
  sending via email 66
Filtering influences 430
Final value results
  arrays 689
  distributions 639
  multi-variate 672
Financial functions 128
Financial Module 53
Finding
  elements 112
  feedback loops 348
  recursive loops 962
Floor function 125
Flows, material and information 150
Forecasting using exponential smoothing 328
Freeing a clone 955
Function elements 69, 147, 274
Function of View 112
SubModels 1009
Functions
absolute value 125
arccosine 124
arcsine 124
arctangent 124
array 801
Bessel 126
beta 126
built-in 124
celling 125
changed 126
constructors for arrays 795
cosine 124
cotangent 124
error 126
exponential 125
financial 128
floor 125
gamma 126
Gauss error 126
hyperbolic cosine 124
hyperbolic sine 124
hyperbolic tangent 124
If then 126
Importance sampling 126
logarithm base 10 125
Math 125
maximum 125
minimum 125
modulus 125
natural logarithm 125
occurs 126
Occurs 417
round 125
sine 124
special 126
square root 125
standard normal 126
Student’s t distribution 126
tangent 124
trigonometry 124
truncate 125
Future value, computing 128

G
Gamma distribution 169, 1061
Gamma function 126
Gauss error function 126
Generic database
creating 1122
downloading from 1035
Global containers 134
Global properties 484
Global Stores for Resources 842
Globalizing a Container 952
Globally downloading from
databases 1040
GoldSim
basic concepts 29
blog 22
calculation logic 340
context-sensitive Help 21
described 27
features 3
file extension 62
installing and activating 7
learning to use 19
license agreement 8
Maintenance program 21
model library 22
modules 52
mouse actions 60
online Help 21
Player 55, 777
referencing time 101, 486
splash screen 58
testing the installation 17
training 22
types of elements 68
user interface described 58
Graphics
adding images 753
adding objects 743
ing objects 746
manipulating 767
Graphics pane
background 423
copying settings 432
customizing 423
defined 59
display tool-tip 61
exporting 772
grid 423
inserting new elements 73
moving objects with mouse 61
navigating within 104
navigation bar 96
printing 771
saving position and scale 426
selecting multiple objects 61
sizing 425
synchronizing with browser 108
zooming 104
Graphics tab
for Containers 139, 1002
Grids
in charts 709
in graphics pane 423
Grouping objects 769
H

Header and footer of charts 704
Help 21
Hiding
   influences 430
      the browser 105, 755
Hierarchical models 39
Hierarchy, creating with Containers 132
High resolution histories
   viewing results 629
History Generator elements 826
   correlating arrays 837
   geometric growth 830
   geometric growth with reversion 833
      random walk 834
      random walk with reversion 835
      reversion to median 833
      reversion to target 835
   types of histories 828
Home button when searching 86
Hyperbolic cosine function 124
Hyperbolic sine function 124
Hyperbolic tangent function 124
Hyperlinks
   adding to graphics pane 760
   appearance 763
   in Notes 759
   in text boxes 753
   locking 761
   making editable 761
   specifying addresses 762

I

If then function 126
   vector/matrix arguments 801
Images
   adding to graphics pane 753
   changing appearance 754
   globally resetting 435
Iman and Conover 1070
ImpOld function 126
Importance measures
   computing 1092
Importance sampling 1079
   custom 1013
   events 366
   Random Choice 375
   Stochastics 181
Importance sampling function 126
Importing
   chart styles 715
   data from spreadsheets 910
   from databases 1031
   text data into 1-D Tables 298
   text data into 2-D Tables 301
   text data into 3-D Tables 304
ImpProb function 126
ImpWeight function 126
Influences
   adding nodes 427
   adding text 428
   between containers 99
   changing appearance of 426
   color 426
   defined 80
   deleting 87
   filtering 430
   globally editing in a Container 428
   hiding 430
   properties 98
   segmented 427
   shape 426
   thickness 426
Information Delay elements 321
   initial values 325
   mathematics of 327
   no dispersion 323
   simulating forecasts 328
   specifying inputs 323
   time-variable delay times 326
   with dispersion 324
Information flow 150
Information tab
   Containers 140
   simulation settings dialog 485
Information tab or SubModels 1002
Input elements 68, 146
Input fields
   auto-complete 83
   auto-suggest 83
   context menu 85
   divide by zero 83
   entering expressions 81
   error checking 93
   specifying contents 79
   tool-tips 111
   using constants in 129
   using functions in 124
Inputs
   entering units 91
   ports 76
Insert Link dialog 85
Inserting
   graphic objects into the graphics pane 743
   images into the graphics pane 753
   new elements 73
Result elements 571
shapes into the graphics pane 743
text boxes into graphics pane 751
text into graphics pane 747
Installing GoldSim 7
Integral of Lookup Table 316
Integration algorithm for Stocks 1137
Integrator elements 228
computing moving averages 233
integration algorithm 229
integration option 232
specifying discrete changes 232
Specifying discrete changes 385
specifying inputs 231
Integration algorithm for Stocks 229
Internal clocks for Containers 136, 474
Interrupt elements 409
buttons on message dialog 412
Continue 412
message 411
Pause 412
triggering interruption 410
without message 414
writing to run log 414
Is_Full output of a Pool 272
Is_Full output of a Reservoir 250

K
Key combinations
conventions to describe 7
Keywords in chart styles 717
Kurtosis of a distribution 1047

L
Labels, changing for elements 436
Last Value tool-tips 557
Latin hypercube sampling 480, 1077
Legends in charts 709
License
agreement 8
Network 13
sharing between versions 19
Standalone 8
Line styles in charts 710
Lines
adding to graphics pane 744
between elements (influences) 98
Link cursor
deactivating 95
described 95
Link types 99, 101
Links
between containers 99
creating 67, 80, 95
deleting 87
Local containers 134
Local Stores for Resources 844
Localized Containers
creating 944
defining output aliases 949
exposing outputs 946
globalizing 952
nesting 948
referencing outputs 945
search logic 951
understanding 944
Locally available properties 808
for Resources 852
Locations
of Resources 855
Locking
Containers 144
Hyperlinks 761
text boxes 752
Locking onto a file
DLLs 936
File elements 943
spreadsheets 926
Logarithm base 10 function 125
Log-cumulative distribution 1058
Logging simulation events 447
Logging simulations 548
Logical elements 290
Log-Normal distribution 1062
Log-Normal Distribution 170
Log-Triangular distribution 1067
Log-Uniform distribution 1068
Lookup Tables 294
controlling interpolation and
extrapolation 312
defining using an external DLL 312
defining using External elements 937
derivative of 1-D 316
dynamic 320
handling data outside bounds 312
importing from a text file 298, 301, 304
integral of 1-D 316
inverse lookup 315
linking to a database 311
linking to a spreadsheet 306
linking to a text file 310
linking to an external data source 305
pasting into 1-D 297
pasting into 2-D 300
pasting into 3-D 304
referencing 314
reverse lookup 315
specifying 1-D tables manually 296
specifying 2-D tables manually 299
specifying 3-D tables manually 302
specifying data 296
steps for defining 294
Looping Containers 137
controlling looping 964
elements 966
using 963
Loops
feedback 346
finding feedback 348
finding recursive 962
recursive 960

M
Maintenance program 21
Manual
list of appendices 6
Notes in 7
organization of 5
Warnings in 7
Material Delay elements 329
applying an inflow limit 334
closing feedback loops 335
initial outflows 333
mathematics of 339
no Dispersion 331
recirculating systems 335
specifying inputs 330
time-variable delay times 333
with dispersion 332
Material flow 150
Math functions 125
Mathematical model 25
Matrices
copying between models 807
creating with constructor functions 795
creating with Data elements 790
creating with Stochastic elements 795
creating with Time Series elements 199
functions that operate on 801
introduction 41
manipulating in expressions 799
manipulating with elements 806
referencing an item 793
using 782
viewing final values 694
viewing in browsers 793
Maximum function 125
Mean time for an Event Delay 379
Menu bars
  docking 423
  main 59
  moving 423
Menus
  context-sensitive 61
  main 59
Microsoft Windows
  versions supported by GoldSim 4
Milestone elements 403
Minimum function 125
Model
  conceptual 25
  mathematical 25
Model author 485
Model Container 95
Model library 22
Model root 95
Modeling aging chains
  Pushed_Out output in Integrators 234
Models
  copying elements between 103
  defining audiences for 738
  example files 5
  hierarchical 39
  navigating 104
  organizing 738
  organizing using Containers 738
  overview of documenting 49
  overview of running 117
Modes
  Edit 498
  overview 498
  Result 504
Modules
  activating and deactivating 18
  Contaminant Transport 53
  Distributed Processing 55
  Financial 53
  overview 52
  Reliability 53
Modulus function 125
Monte Carlo options
  # of realizations 479
  sampling method 480
  setting 479
Monte Carlo result options 482
Monte Carlo simulation
defined 1050
overview 37
using distributed processing 55
Mouse actions
conventions to describe 7
defined 60
displaying context-sensitive menu 61
displaying tool-tip 61
dragging 61
dragging objects 61
dragging objects precisely 769
dragging toolbars and menu bars 423
Moving
browser 105, 755
elements between containers 102
objects 61
objects precisely 769
toolbars and menu bars 423
Moving averages 233
Multi-variate results
properties 675
Multi-variate results 672
classifying results in a scatter plot 686
controlling chart style 688
defined 561
exporting 733
reassigning axes 674
screening results 686
selecting outputs for display 672
sensitivity analysis 681
viewing a 2D scatter plot 677
viewing a 3D scatter plot 679
viewing a correlation matrix 683
viewing data 684

N
Naming elements 78
Natural logarithm function 125
Navigating models 104
Navigation bar in graphics pane 96
Negative binomial distribution 170, 1063
Nested Monte Carlo 992
Net present value, computing 128
Network license 13
borrowing 15
connecting to 13
Normal distribution 171, 1063
Not elements 293
Notes
creating 758
deleting 758, 759
described 755
editing and formatting 759
inserting hyperlinks 759
viewing the note pane 755
NPV, computing 128
Number in queue 381
Numerical integration 1137
NumOfReal 492

O
Objective function for optimization 530
Objects
aligning and ordering 767
copying and pasting 770
displaying tool-tips 61
graphical 60
graphics and text 742
grouping 769
model 60
moving precisely 769
moving with mouse 61
rotating 769
selecting multiple 61
selecting with mouse 60
spacing and sizing 768
Occurs function 126, 417
Offsets
for Spreadsheet elements input and outputs 923
Opening files 62
Optimization 528
objective function 530
optimization variables 531
overview 528
precision 534
probabilistic 538
randomizing 534
required condition 530
running 535
saving settings and results 538
settings 529
warning messages 537
Optimizing a SubModel 1003
Options
updating expressions 103
Options dialog
General tab 446
Graphic tab 447
Results tab 448
Or elements 292
order of calculation for elements 968
Ordering objects 767
Organization of manual 5
Organizing your model 738
Oscillatory behavior
  Pools 269
  Reservoirs 248
Outflows from Pools 265
Outline, changing for element 437
Output attributes 87
Outputs
  continuous and discrete 89
  discrete signals 89
  highlighting as saved results 497
  ports 76
  saving as results 494
Overflow rates from Pools 260
Overflow rates from Reservoirs 242

P
Pareto distribution 1064
Pareto Distribution 172
Partial correlation coefficients
  computing 1091
Password
  for locked Container 144
Pasting
  data into a 1-D Table 297
  data into a 2-D Table 300
  data into a 3-D Table 304
  data into a Time Series 197
  data into an array 793
Pausing a model 502
PDF 1045
  computing 1085
Pearson Type III distribution 1064
Pearson Type III Distribution 172
Percent unit 446
Percentages 446
Percentiles
  selecting for probability histories 601
  viewing time histories of 600
  when defining distributions 156
PERT distribution 161
Player 55, 777
Plot points, saving 464
Plots, see Charts 564
Poisson distribution 1065
Poisson Distribution 173
Poisson events 1083
Polygons, adding to graphics pane 744
Polylines, adding to graphics pane 744
Pool elements 252
comparing individual outflows 265
computing overflow rates 260
computing total outflow 263
Inflows tab 254
integration algorithm 257
Is_Full output 272
oscillations 269
Outflows tab 256
replacing current value 273
specifying discrete changes 273
Specifying discrete changes 388
specifying units 254
upper and lower bounds 258
Ports
  defined 76
  hiding 76
  options for display 76
Precedence
  for mathematical operators 81
  for relational operators 130
Precedence conditions in Triggers 361
Precision
  of saved results 569
  of tool-tips 112
Present value, computing 128
Previous button when searching 86
Previous Value elements 957
browser view 960
creating recursive loops 960
inputs and outputs 958
using to solve coupled equations 1139
Printing
  graphics pane 771
Probabilistic simulation 24
  options 479
  using Stochastic elements 153
Probabilistic simulations
  compared to deterministic 1051
Probability density function 1045
Probability histories 600
  selecting percentiles 601
Properties
  displaying 61
  of elements 77
  of graphic objects 746
  of Hyperlink objects 762
  of influences 98
  of text boxes 752
  of text objects 748
Properties of results 564
Protecting a model file 65
Protecting Containers 139
Push Instruction
for modeling aging chains 814
in Discrete Change element 384
use in Integrators 234

Q

Queues
Discrete Change Delays 396
Event Delays 381

R

Random Choice elements 373
importance sampling 375
Random events
mathematics 1083
Random events, generating 365
Random number seeds 481, 1072
Randomizing optimization sequence 534
Realization weights 481
Realizations
classifying 577
defined 37
displaying time histories of 597
referencing current number 492
running one at a time 502
screening 577
simulation options 479
Recovering files after crash 63
Rectangles, adding to graphics pane 745
Recursive loops 960
finding 962
Redo 771
Reference version 1025
Regular events, generating 365
Relational operators 130
Reliability Module 53
Removing units 93
Reporting periods
defining 461
viewing results 607
ReportingMonth 493
Required condition, in Triggers 362
Requirements
operating system 4
Reservoir elements 235
computing overflow rates 242
computing withdrawal rates 245
integration algorithm 237
Is_Full output 250
oscillations 248
replacing current value 251
specifying discrete changes 251
Specifying discrete changes 386
specifying primary inputs 237
upper and lower bounds 240
withdrawal rate output 238
Resetting
element images globally 435
Resources
allocating amongst competing requirements 851
creating and editing Global Stores 842
creating and editing Local Stores 844
defining and editing Resource Types 840
defining requirements 848
elements that interact with 850
for Discrete Change Delays 396
for Event Delays 382
generating 855
histories 859
in SubModels 1001
interacting with 848
introduction 839
introduction to creating 839
locally available properties 852
locations and users 855
moving between Stores 853
referencing resource availability and use 852
results 859
Result
properties 564
Result array, for distributions 653, 1084
Result display
charts 564
dialogs 564
overview 556
precision 569
properties 564
size of values 569
tables 564
tool-tips 557
types 558
Result elements
creating 571
defined 149
described 570
disabling 635
exporting to spreadsheets 720
exporting to text files 727
finding connected outputs 574
special uses 576
types of 72
viewing 573
Result Mode 504
Results
archiving 497
arrays 689
classifying distribution results 658
classifying realizations 577
classifying time history results 615
disk space required 496
displaying multiple distributions in a chart 654
displaying multiple time histories in one chart 588
displaying single realizations 658
distribution 639
exporting 719
globally saving using Containers 141
highlighting saved 497
introduction to viewing 118
multi-variate 672
overview of display 556
saving for specific outputs 494
screening array results 699
screening distribution results 658
screening multi-variate results 686
screening realizations 577
screening time history results 615
specifying how saved 494
time history 581
types of 558
Reversion to median
History Generator elements 833
Reversion to target
History Generator elements 835
Rotating objects 769
Round function 125
row
use in array constructors 795
RTime 490
Run Control toolbar
slowing down a simulation 503
using 497
Run log 548
Run Mode
aborting a run 503
pausing a model 502
slowing down a simulation 503
states 500
stepping through a model 502
Run properties 486
Running a model
details 497
from command line 549
overview 115, 117
using a batch file 549
viewing results while 574
S
Sampled result distribution 1065
Sampled Results distribution 174
Sampling method
importance sampling 181
Latin hypercube 480
Sampling Stochastic elements 177
Saving
chart styles 712
files 62
graphics pane settings 426
Player files 777
results for specific outputs 494
results for subelements in Containers 141
results globally using Containers 141
Saving an inventory file 772
Scatter plots
2D 677
3D 679
classifying results 686
Scenario Data 507
Scenario Manager 513
Scenario Mode 520
from Result Mode 526
Scenarios
Basic Concepts 505
defined 505
exporting time histories to spreadsheets 725
Overview 504
overview of displaying results 577
running and displaying results 520
viewing distribution results 665
viewing time history results 622
Scientific notation 448
Screening realizations 577
Script elements
assigning values to variables 864
break statements 884
breakpoints 890
browser view 898
comments 890
continue statements 884
controlling program flow 874
debugging 890
defining and assigning arrays 870
defining local variables 867
defining outputs 872
deleting statements 888
do loops 879
documenting 890
editing 887
examples 895
for loops 876
getting started 862
hot-keys 887
if-else statements 874
inserting statements 863
introduction 860
keyboard shortcuts 887
log statements 892
moving statements 889
outputs 872
printing 894
repeat until loops 881
scope of variables 885
Variable Assignment statement 864
Variable Definition statement 867
while loops 883
Sealing Containers 143
Searching for elements 112
Seeds, random number 481, 1072
Selecting
  multiple objects 61
  objects with mouse 60
Selector elements 278
Sending a model via email 66
Sensitivity analysis
  central value result 544
  graphical 539
  multi-variable 681
  single variable 539
  statistical 681
  tornado chart 544
  X-Y function charts 545
Sequencing logic 968
Shift button
  Spreadsheet elements 925
Shorter timestep periods 467
Show Changes (between versions) 1027
Show element ports 76
Show element subitems 109
Significant figures 448
Simple database
  creating 1122
  downloading from 1037
Simulated Bayesian updating 1016
Simulation
  accuracy of models 1136
  basic concepts 24
  dynamic 24, 34
  probabilistic 24
  static 455
Simulation modes
  defined 66
  described 498
Simulation settings
  advanced timestep options 466
  analysis description 485
  described 452
  deterministic runs 482
  dynamic model 455
  for a SubModel 978
  globals 484
  information 485
  model author 485
  Monte Carlo options 479
  Monte Carlo sampling method 480
  overview 115
  probabilistic runs 479
  time options 453
Sine function 124
Sizing objects 768
Skewness of a distribution 1047
Slowing down a simulation 503
Sorting table results 568
Spacing objects 768
Special functions 126
Splash screen 58
Splitter elements 281
Spreadsheet elements 910
  browser view 931
  controlling when it links to spreadsheet 927
  creating and editing inputs 913
  creating and editing outputs 916
  creating inputs using the wizard 914
  creating outputs using the wizard 921
  defining offsets 923
  defining properties 911
  exchanging dates 930
  exporting data to the spreadsheet 913
  importing data from the spreadsheet 916
  importing data in Edit Mode 929
  importing probability distributions 918
  locking onto a file 926
  saving changes to spreadsheet 929
  saving outputs 930
  shifting ranges simultaneously 925
Update Spreadsheet Outputs
option 929
wizard for exporting data to the spreadsheet 914
wizard for importing data from the spreadsheet 921

Spreadsheets
exporting results from Time History Result elements 720
exporting results using Spreadsheet elements 732
exporting scenario results from Time History Result elements 725
linking to Lookup Tables 306
linking to Time Series elements 194
linking to via Spreadsheet elements 910

Square root function 125

Standalone license
activating 8
deactivating 10
reactivating 12
Reactivating 12
transferring 10
upgrading 12

Standard deviation of a distribution 1047
Standard normal function 126
Standard regression coefficients computing 1091
Standard toolbar 59
Start Dialog 58
State variables 341
Static simulation 455
Status bar 59
Status elements 398
representing a deadband 400

Stepping through a model 502
Stochastic elements 153
autocorrelating 179
controlling sampling 407
correlating 178
defining as vectors 183
defining distributions 155
deterministic value 180
distribution functions 182
Distribution output 155
distribution types 158
importance sampling of 181
saving results for 155
triggering 407

Stock elements
defined 147
types of 69

using 228
Stores
Global 842
Local 844
Student’s t distribution 174, 1065
Student’s t distribution function 126

Style manager 714
Styles for charts 581
Subelements
saving 141
Submodels 973

SubModels
Affects View 1009
appearance of contents 1002
applications of 973
building contents 977
controlling when run 990
creating 974
examples 1011
exporting 1006
Function of View 1009
importing 1007
Information tab 1002
input interface 980
interrupting 1009
locking 1005
logging run messages 1005
modules 976
Nested Monte Carlo 992
optimizing 1003
output interface 984
pausing 1009
protecting contents 1005
Resources 1001
saving results inside 991
sealing 1005
simulation settings 978
solution type 976
viewing dependencies 1009
viewing histories in parent model 617
viewing results inside 991
why use 973

SubSystems
defined 135
in feedback loops 348

Sum elements 289
Support
e-mail 21
GoldSim user forum 21
model library 22

Symbols for elements
changing 434
globally changing 435
Synchronizing graphics pane and browser 97

T

Table results
  distributions 651
  selecting and copying 567
  size of values displayed 569
  sorting 568
  time histories 586
Table Results 564
  copying 734
Tables
  Lookup 294
Tangent function 124
Technical support 21
Temperature units 92
Termination event
  output of conditional Container 904
Testing installation 17
Text
  adding to an influence 428
  adding to graphics pane 747
  changing appearance in graphics pane 748
  editing in graphics pane 748
  pasting with tabs in graphics pane 770
Text boxes
  inserting hyperlinks 753
Text boxes
  adding to graphics pane 751
  changing appearance in graphics pane 752
  editing in graphics pane 752
  unlocking in graphics pane 752
Text files
  exporting results from Time History Result elements 727
  Time
    options 453
    referencing in GoldSim 101, 486
  Time basis for simulation 453
  Time history results 581
    arrays 592
    classifying results 615
    controlling chart style 635
    multiple realizations 597
    multiple realizations for multiple outputs 605
    multiple realizations of an array 606
    plotting conditions 585
    probability histories 600
    properties 581
    reporting period-based 607
    screening results 615
    viewing charts 584
    viewing custom statistics 603
    viewing multiple outputs 588
    viewing percentiles 600
    viewing scenarios 622
    viewing SubModel histories 617
    viewing tables 586
    viewing Time History Definition outputs 617
    viewing unscheduled updates 629
  Time History results
    defined 559
    exporting results to spreadsheets 720
    exporting results to text files 727
  Time Series elements 186
    advanced options 213
    browser view 227
    changes over intervals 209
    constant values over intervals 206
    data source 188
    defining discrete changes 188
    defining multiple time series data sets 218
    defining outputs 200
    defining vector data 199
    discrete changes 211
    elapsed time entries 197
    examples 204
    generating discrete changes 415
    importing data from spreadsheets 194
    input data represents 190
    instantaneous values 204
    linking to a time series definition 225
    pasting data into 197
    referencing as a function 203
    referencing dates 197
    shifting origin of data 214
    time shifting data 214
    type and units 188
    using to record 221
    viewing and editing data 191
  Timed Event elements 365
    importance sampling 366
    mathematics 1083
    updating between timesteps 367
  Timestepping algorithm 1140
  Timesteps
    advanced options 466
    alignment 459
    capture points 469
controlling unscheduled updates 470
customizing 466
dynamically adjusting 471
Elapsed or Calendar 453
internal for Containers 474
length 459
overview 455, 470
pausing model between 502
periods with shorter 467
plot points 464
referencing length 492
reporting periods 461
saving 464
selecting 1140
Toolbars
activating and deactivating 422
customizing 422
docking 423
Drawing Tools 742
inserting elements 73
moving 423
Run Control 59
standard 59
types 422
Tool-tips 109
displaying 61
for elements 109
for input fields 111
in chart display windows 567
viewing large and small numbers
in 112
viewing results 557
Top-down models 39
Tornado chart 544
Total outflow from Pools 263
Tracking model changes 1023
Training course 22
Triangular distribution 175, 1066
Triggered Event elements 368
specifying resources 369
Triggering
defined 355
defining triggering events 356
multiple elements simultaneously 905
precedence conditions 361
required condition 362
Resource Interactions 364
Stochastics 407
Trigonometry functions 124
Truncate function 125
Type button in properties dialogs 88
Types of elements 30
Types of links 99, 101

U
Uncertainty
differentiating from variability 1049
propagating 1050
quantifying 1044
representing 36
types 1044
Undo 771
Uniform distribution 176, 1068
Units
appendig automatically 92
casting 93
creating 441
creating for items (e.g., widgets) 444
currencies 92
display 89
entering 91
for dimensionally inconsistent
expressions 93
Manager 439
managing user-defined 444
overview 89
percentage symbol 446
removing 93
rules for entering 90, 91
saving settings 444
table of conversion factors 1110
temperature 92
Units.dat file 444
Unlocking
Containers 144
Hyperlinks 761
text boxes 752
Unscheduled updates
controlling 470
defined 458
saving results 477
viewing results 629
User interface
browser 59
described 58
graphics pane 59
menu bar 59
Run Control toolbar 59
standard toolbar 59
status bar 59
toolbars 422
Users
of Resources 855

V
Validating database links 1035
Variability
differentiating from uncertainty 1049

Variance of a distribution 1047

Vectors
copying between models 807
creating with constructor functions 795
creating with Data elements 790
creating with Stochastic elements 795
creating with Time Series elements 199
functions that operate on 801
introduction 41
manipulating in expressions 799
manipulating with elements 806
referencing an item 793
using 782
using as lookup tables 798
viewing final values 692
viewing in browsers 793

Version Manager 1025

Versioning
changes tracked 1026
creating versions 1024
defined 1023
deleting versions 1025
disabling 1026
displaying differences 1027
enabling 1024
overview 1023
reference version 1025
reports 1030
Version Change Note 1029
Version Manager 1025

Viewing
element dependencies 112
results, see Displaying results 556

Views
Affects 112
Functions Of 112
types of in browser 107

W
Weibull distribution 176, 1069
Withdrawal rates from Reservoirs 238, 245

X
XML model inventory 772

Z
Zooming
in charts 567
in graphics pane 104